# Modicon TSX Quantum
140 ERT 854 10
User manual

840 USE 477 00 eng

33000621.01

# Table of Contents

# About the book

## At a Glance

**Document Scope**

This manual describes the mode of functioning and performance of the 140 ERT 854 10 module as well as the usage with the TSX Quantum. It should show you how to provide your Quantum with time stamped data.

**Validity Note**

The information in this document is valid for concept version 2.2 and later.

**Related Documents**

| **Title of Documentation** | **Reference Number** |
|---|---|
| Concept User Manual | 840 USE 493 00 |
| Quantum Hardware User Manual | 840 USE 100 00 |
| DCF 077 Benutzerhandbuch | 840 USE 470 02 |
| COM ESI Planning Manual | 840 USE 458 00 |
| PRO TSX 101 Planning Guidelines | 840 USE 476 00 |

**User Comments**

We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

# Function Overview

I

## Introduction

**Overview**

The first part of the manual for the intelligent input module 140 ERT 854 10 gives an overview of the structure of the module, the functionality and shows typical applications.

**What's in this part?**

This Part contains the following Chapters:

# Introduction

1

## Module Overview

**Overview**

The 140 ERT 854 10 is an intelligent 32 point input module for TSX Quantum that allows full configuration of inputs and evaluates the input signal status every 1 millisecond. Up to 9 ERTs can be installed on a local or remote module rack and can be used with Concept versions starting from V2.2.

**The Inputs**

The 32 inputs are designed for input voltages of 24 to 125 VDC and are distributed in 2 independent groups. Each group is supplied with a separate external reference voltage (typically 24, 48, 60 or 125 VDC), to influence the threshold limit and minimum current consumption. The module status Ready, Active and Error as well as the input status (status of the terminals) are clearly displayed by the status LEDs on the module.
140 ERT 854 10 firmware processes inputs in four separate configurable function blocks with 8 inputs which support the following functions that can be selected.

- Binary inputs: input values are sent cyclically to the PLC.
- Event inputs: Time registered event logging for 1, 2 or 8 processed inputs, with 5 byte time register, integrated FIFO buffer for 4096 events and acknowledging PLC transfer by the user.
- Counter inputs: 32 bit addition of processed events up to 500 Hz that are transferred cyclically to the PLC.

Parameters can be set for processing individual inputs: (disabled, inverted, and with debouce filter). A configurable chatter filter can be activated for the event and counter inputs and event edge monitoring carried out.

**Time Synchronization**

The module clock requires a time synchronization signal and provides a 24 VDC input with potential isolation for the following standard time receiver with DCF 77 format.
- DCF 77E (long wave reception only in Europe)
- 470 GPS 001 (Global satellite receiver)

The ERT internal software clock can alternatively be created by the application program, or be free running.

**Validity reserve**

A validity reserve can determine how long the module clock can continue running without external synchronization. The ERT data evaluated can be buffered with a maximum current consumption of 0.07 mA by the 140 XCP 900 00 battery module in the event of power loss.. The current internal software time is transferred to the PLC at proportional intervals and enables the CPU clock to be set by the application program.   For further information see *Time synchronization with standard time, p. 21*.

# User Functions and Services

2

## Introduction

**Overview**

The 32 inputs of the 140 ERT 854 10 module can be individually preprocessed and transferred to the PLC as binary value, counter value or event. The following chapter describes the functions and services available.

**What's in this Chapter?**

This Chapter contains the following Maps:

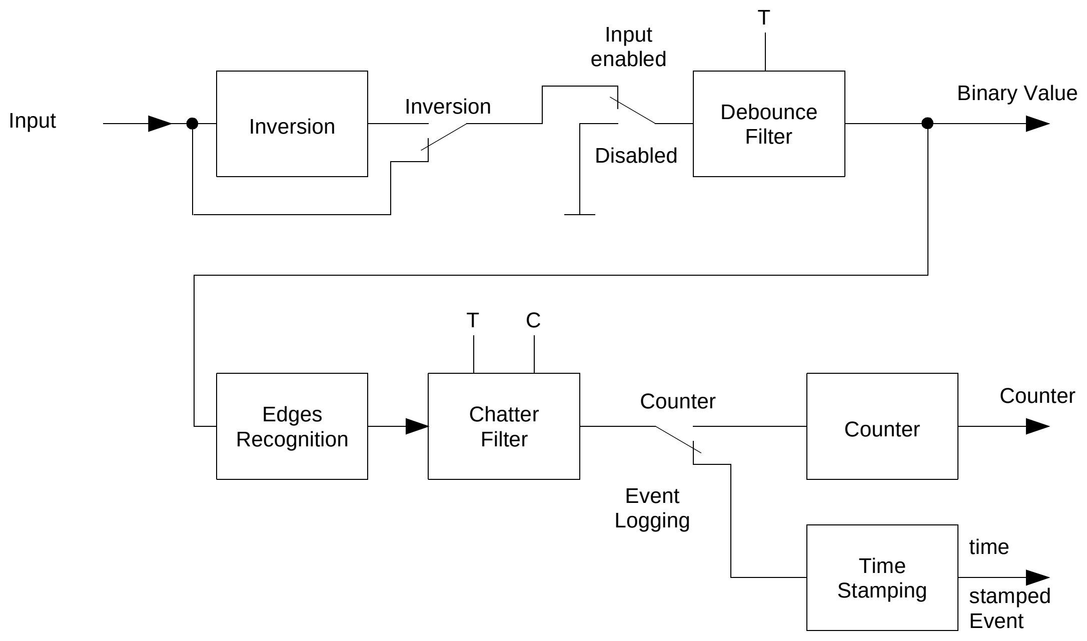
## Input Processing - Registration and Filtering

**Overview**

The input signals connected to the 140 ERT 854 10 go through a multistage processing before that are available to the user program as binary, counter values or events. The processing can be set with parameters for each individual input.

**Signal Processing Sequence**

The processing of the input signals is done according to the parameters set. Parameters are set on the Parameter Screen in Concept I/O Configurator.



## Registration

| | |
|---|---|
| **Overview** | The processing of the individual inputs is completely configurable: (disabled, inverted and with debounce time). The event inputs can also have a configurable chatter filter activated and an edge event evaluation. |
| **Disabling** | A disabled input always shows the value "0" independent for its input state |
| **Inverting** | The input polarity is inverted before further processing. If this is active, the opposite to the input signal status shown on the status LEDS is passed on for further processing. |
| **Edge Recognition** | Selects the edge transitions which should be used for active events and counter inputs. "Both Edges" processes rising and falling edges. Otherwise only a signal edge is processed: rising/falling, either with or without active inversion. |

## Filtering

**Overview**

The configurable filtering is done in 2 stages: debounce and chatter removal.

| | CAUTION |
|---|---|
| ⚠ | **Danger of incorrect interpretation of the input data**<br>Filters are used to suppress the input recognition in a defined way. Filtering should only be used in a suitable way to prevent too much or undesired suppression of input data.<br>**Failure to observe this precaution can result in injury or equipment damage.** |

**Debounce**

Debouncing can be used on all input functions and prevents the processing of fast state changes of the inputs, like for example, those caused by contact bouncing. Signal changes are ignored depending on the filter type and the preset time. The value range for the filter time is 0 to 255 ms; the value 0 deactivated the debounce filter. The selection of the debounce filter type "stable signal" or "integrating" affects all 8 function block inputs.

- "Stable Signal" Filtering: A signal change is only registered if the polarity change stays stable for longer than the filter time (each new change resets the filter time).
- "Integrating" Filtering: A signal change is only registered if the time integral of the input signal reaches the programmed filter time taking any polarity change into account.

**Note:** Debounce Time>=1 ms is recommended to ensure enough immunity against electromagnetic disturbances. This means that input signal states >= 2 ms and events up to 250 Hz can be processed. In non-critical electromagnetic environments, the debounce time can be set to 0 to avoid unnecessary filter delays. This means that input signal states >= 1 ms and events up to 500 Hz can be processed.

## Chatter Removal

Chatter removal can only be used for event and counter inputs. It limits the number of events to a configurable value during a configurable time period. This should prevent multiple event registrations for the same input, e.g. disturbance influences due to slowly changing inputs (because the hysteresis is possibly set to small). The chatter counter is configurable for each individual input, the chatter time for each input pair. The selection of "chatter removal" on the parameter screen activates the chatter filter for all 8 function block inputs. The chatter filtering for individual inputs can always be disabled by selecting the value of 0 as chatter count value. A "Chatter Filter Active" bit within the "status" output word (Bit 7 - DC) which is returned from the transfer EFB "ERT_854_10" indicates that at least one "chatter" input is being filtered (see *ERT_854_10: Data transfer EFB, p. 69*). The bit is reset as soon as the chatter time of the last active filtered input has run out.

- Chatter Time: The time period in which the chatter count limit has an effect. Value range from 1 ... 255 * 100 ms = 0.1 ... 25.5 Seconds.
- Chatter Count: The maximum number of registered events which are allowed to be passed on within the chatter time period. Value range from 1 ... 255, the value 0 deactivates the chatter filter.

**CAUTION**

**Danger of incorrect interpretation of the input data**

The chatter removal is a very powerful processing tool which can have undesired side effects. Its use with counter inputs is questionable. If edge recognition is performed for "Both Edges" then, in the case of odd-numbered chatter suppression, two successive events with the same edge (2 rising, 2 falling) appear when transferred to the PLC.

**Failure to observe this precaution can result in injury or equipment damage.**

## Input Data Processing

**Overview**

The input signal can be used as binary inputs, counter values or for event recording depending on the parameters set in the concept I/O Paramter dialog box.
Normally the input data of the ERT 854 module is processed by the corresponding EFBs (see *EFBs for the140 ERT 854 10, p. 61*)

**Binary Inputs**

All inputs of the function block are transferred to the PLC after the third processing stage (i.e. enabling, inverting and debounce filtering) before the chatter filter and edge recognition are performed. The processed values of all 32 inputs are cyclically transferred (every second PLC cycle) to that first and second input register word of the7 word 3x register block of the ERT The address sequence of the module inputs corresponds to standard digital input modules, i.e. inputs 1 ... 16 correspond to bits 15 ... 0) The confirmation by the user is not necessary because the EFB ERT_854_10 must exist and be enabled. The processed values are available for all 32 inputs independent of the further processing as counter or event counter. The input processing is always done according to the configuration, but the ERT copies the processed values from the input immediately after the third input processing stage !

**Note:** If the BoolArr32 output array "Input" of the "ERT_854_10" transfer EFB has been configured (see *ERT_854_10: Data transfer EFB, p. 69*) , the processed values are available as boolean values.

## Counter Values

All inputs of the function block go through all five input processing stages (i.e. enabling, inverting, debounce and chatter filtering as well as edge recognition). The counting is done as soon as the edge reaches the edge recognition. For edge recognition which is not set as "both edges", the configured inverting decides if rising or falling edges are counted.

**Note:** Inversion is probably not sensible to use with the recognition of "both edges"

Counter values are 32 bit totals. The PLC receives a complete sequence (configured as 8, 16, 24 of 32) of time consistent counter values in a multiplex procedure for the "ERT_854_10" transfer EFB (see description of the EFB, section *EFBs for the140 ERT 854 10, p. 61*). The EFB sets the values in the configured UDINTArr32 output array"Cnt_Data", without the confirmation of the user. After the transfer of the new counter values is completed, the EFB sets the signal "new data", a boolean variable from "ND_Count", for one PLC cycle.

**Note:** The transfer of the counter values starts with function block 1 and ends with the last function block which is configured as counter inputs. If a consecutive sequence of function blocks starting with the first block are configured as counter inputs, transfer resources are saved. Since the transfer of the counter values competes with the transfer of the recorded events, faster reactions time for both types can be achieved if an ERT module is completely configured as either counter or completely as event inputs. Binary and condition inputs have no effect on this.

**Event logging**

This function allows input state changes to be registered in time order with a high resolution. The input state changes are logged with a time stamp with high resolution. The events can later be shown in the correct sequence. The time stamp logging of events can be configured so that a group of 1, 2 or 8 inputs can be processed in parallel. ALl inputs of the function block go through all five input processing stages (i.e. enabling, inverting, debounce and chatter filtering as well as edge recognition). The logging (including time stamping) is done as soon as the edge reaches the edge recognition. For edge recognition which is not set as "both edges", the configured inverting decides if rising or falling edges are logged.

**Note:** Inversion is probably not sensible to use with the recognition of "both edges".

A group of inputs is logged as an event if at least one of the inputs in this group has an edge which has been recognized, i.e.:
- any single input (1, 2 ... 7, 8),
- any input of an input pair (1-2, 3-4, 5-6, 7-8),
- an input of an 8 bit group.

Events contain a lot of information in an 8 byte block, including the processed values of all inputs in the group with the corresponding time stamp:
- Module Number
- Type of input group and number of the first bit
- The current value of the inputs in the group
- Time Stamp: Milliseconds
- Time Stamp: Minute
- Time Stamp: Hour
- Time Stamp: Day of the week / Day in the month

The current value of the inputs is stored right justified in an event structure byte. Teh ERT saves up to 4096 events in its battery backed FIFO buffer. The ERT provides error bits (bit 5/6 - PF/PH) for buffer overflow/Buffer half full within the "status"output word which is returned from the "ERT_854_10" transfer EFB. Individual events are transferred in a "ERT_10_TTag" structure on the PLC by the "ERT_854_10" transfer EFB. After processing the events, the user must actively signal readiness for the receiving of new events.  See EFB Description *ERT_854_10: Data transfer EFB, p. 69*. If desired, the parameter "complete time report" can be selected to provide the month and year. For this purpose, there is a special pseudo event without values which contains the complete time information with month and year. The event is marked as a "complete time report" and precedes the "actual", time stamped event. (See additional information about "Complete time Report" in *Parameter and Default values, p. 45*).

## Status Inputs

**Status word**

The "status" output word which is cyclically returned by the "ERT_854_10" transfer EFB contains the following error bits:
- D8 ... D0 ERT error bits
- D11 ... D9 reserved
- D15 ... D12 EFB error bits

A complete description of the error bits is in the *Assignments of the Error Bits, p. 78*
After the transfer of the new status inputs is completed, the EFB sets the signal "new data", a boolean variable from "ND_Stat", for one cycle.

**Note:** ERT/EFB error messages are displayed in the Concept screen **Online** → **Event Display** with error number and explanation (see *Online error display, p. 80*).

# Time Synchronization

# 3

## Time synchronization with standard time

**Overview**

The time stamped event logging requires a precise internal clock. The ERT module uses a software clock for creating the time in millisecond intervals. This software clock is normally synchronized with the help of an external time signal (standard time receiver) in one minute intervals. It can also be synchronized via a telegram or be free running.

The incoming time signal is checked for plausibility. Runtime deviations from the software clock are corrected. The time reception takes a few minutes before the time becomes available after startup. The software clock is synchronized to this time. The module then determines the deviation from the software clock with regard to the external clock within a specific period, and offsets the deviation accordingly. This is carried out continuously during the entire runtime. After a few hours runtime (generally within 2 hours) the software clock reaches maximum precision.

If implausible or incorrect time messages are received, the software clock continues running without sychronization. The deviation gets larger during this time. If this time phase does not exceed the "Power Reserve" specified, the clock resynchronizes when the next valid time information is received.  However, if the time period is exceeded before the module receives a valid time signal, the ERT sets bit "Invalid Time" in the "Status" output word (bit 3 - TU), returned by the "ERT_854_10" transfer EFB (see *ERT_854_10: Data transfer EFB, p. 69*). All time stamps set after this are invalid (the high priority byte for millisecond information is set to FF). The bit is reset as soon as the next valid time message is received.

If the module receives no valid time messages for 10 minutes, the ERT sets the bit "Time Reference Error" in the "Status" output word (bit 2 - TE), returned by the "ERT_854_10" transfer EFB (see *ERT_854_10: Data transfer EFB, p. 69*). The bit is reset as soon as the next valid time message is received.

**Synchronization**

There are three types of synchronization available:

- DCF 77E reception module (German standard - long wave reception only in Europe)
- 470 GPS 001 00 satellite receiver, DCF77 formated signal given (global satellite reception)
- Synchronized by the PLC using "ERT_854_10" EFB (low precision)

**DCF Time base**

The DCF 77E receiver delivers a 24VDC signal in DCF77 format and can supply up to 16 ERT modules concurrently. The BCD coded time signal is transferred once a minute and synchronizes the ERT minutes changeover. When the ERT is restarted the software clock is synchronized within three minutes of receiving the first information. After this the ERT software clock time matches the standard time sender. If the send signal becomes unavailable the free running software clock can still be used but is not as precise. The DCF sender delivers CET (Central European Time), takes into account summer/winter time changes as well as transition seconds and leap years.

**GPS Time base**

A GPS receiver such as the 470 GPS 001 must be used for applications which use GPS satellite time references. This module demodulates the GPS signal and delivers DCF77 format output signal from 24 VDC. The ERT decodes the signal and synchronizes the minutes transition for the internal software clock. GPS satellites sends UTC time (Universal Time Coordinated) which GMT (Greenwich Mean Time = Western European Time) corresponds to. Seconds and years transitions are taken into account. Depending on the location, the local time relative to GMT as well the local summer/winter time changes can be configured with the 470 GPS 001 receiver. The recommended power reserve for the DCF/GPS time base signal is one hour (the settings range for DCF/GPS sync is between 1 ... and 5 hours). Several ERT module groups can be synchronized simultaneously using a GPS receiver. Further information can be found in the manual for 470 GPS 001 00 Recievers.

**EFB synchronized internal clock**

If a clock only requires a lower precision, the ERT internal software clock can be synchronized with a time value sent by the master. The software clock runs freely until the next time value is received. Precision is usually within 100 milliseconds per hour and the software clock must be synchronized correspondingly often. The "ERT_854_10" transfer EFB provides the required time synchronization. This means several ERT modules can be supplied with approximately the same time; the time source used is the ESI 062 00 module data structure "DPM_Time". The power reserve settings for the EFB synchronized internal software clock moves in the range between 1 ... and 254 hours). However, if the time period is exceeded before the next transfer of a time signal, the ERT sets bit "Invalid Time" in the "Status" output word (bit 3 - TU), returned by the "ERT_854_10" transfer EFB. All time stamps set after this are invalid (the high priority byte for millisecond information is set to FF). The bit is reset as soon as the next valid time message is received.

**Free running internal clock**

The ERT internal software clock can also be used on its own. Setting the power reserve for the internal software clock to 0 activates duration mode, shown by the bit "Time not synchronized" in the "Status" output word (bit 4 - TA) which is returned by the "ERT_854_10" transfer EFB. In this case there is no power reserve that can be exceeded and therefore no invalid time stamps. The bits "External Reference Error" and "Invalid Time" in the output word "Status" (Bit 2/3 - TE/TU) are never set; the time starts automatically without synchronization. The default start settings for the internal clock is 0 hours, 1/1/1990. The time settings can be made through:

- a telegram (e.g. by IEC 870-5-101)
- the CPU clock (using the "DPM_Time" data structure).

**Note:** Using the free running internal software clock enables even more precise processing of events within an individual ERT.

# Typical Application Areas

# 4

## Typical areas of application

**Overview**

The ERT 854 10 is particularly suited for determining the binary input status and counter value that require a time stamp

**140 ERT 854 10 Applications**

The following areas of application are valid for the 140 ERT 854 10:

- Processing binary inputs:   Use as a standard I/O module with filtering and an input range of 24 - 125 VDC.
- Event Logging:   The event of an individual process status can be logged with the corresponding time (time stamp). This enables the later reconstruction of the time point and the sequence of process signals "coming" or "going".
- Counter value:   Use as a standard I/O module (with filtering, 32 bit summing with max. 500 Hz) with an input range of 24 - 125 VDC.
- Periodic time stamping of process values:   Recording counter values in defined time intervals. The combined use of both function groups can be used as an advantage here.
- Time dependent switching actions:   Outputs can be set regardless of time for contolling lighting, heating, ventilators, temperatures (building automation), or for opening/closing doors, machines, ... (safety measures). The output status can be recorded with the ERT.

# Module Description

II

## Introduction

**Overview**

The 140 ERT 854 10 is an intelligent digital input module for evaluating input values with or without event recording.

**What's in this part?**

This Part contains the following Chapters:

# Module Description

5

## Introduction

**Overview**

This chapter provides information about the structure of the 140 ERT 854 10 module and its technical data.

**What's in this Chapter?**

This Chapter contains the following Maps:

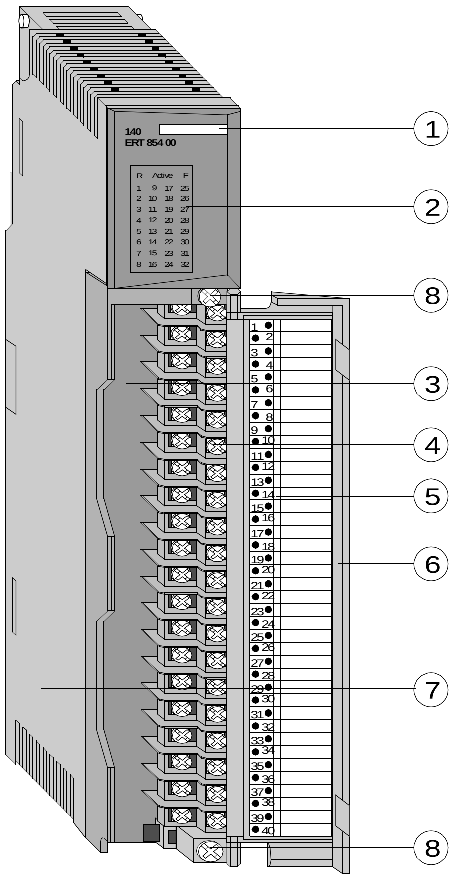
## Overview

**Introduction**

The 140 ERT 854 10 is a TSX Quantum Expert Module with 32 binary inputs (24 ... 125 VDC). The module is suitable for the evaluation of digital inputs, counter pulses and events.

## Front View of the Module

Front View of the ERT 854 10



**Location of Operating Elements**

| | |
|---|---|
| **1** | Color Code |
| **2** | Display field (LEDs) |
| **3** | Terminal Block |
| **4** | Connection terminals |
| **5** | Sliding Label (inside) |
| **6** | Cover for the terminal blocks |
| **7** | Standard housing |
| **8** | Screws for terminal block |

## Features and Functions

**Features**

The ERT 854 10 is a TSX Quantum Expert Module with 2 groups of 16 binary inputs (24 ... .125 VDC). The input groups are potentially isolated to each other and to the internal logic. In addition to the counter pulses, digital input values with our without event recording can be evaluated. A time receiver can be connected for synchronization of the time.

**Mode of Functioning**

The registers of the ERT 854 10 count impulses with frequencies of up to 500 Hz with an interruption/impulse period of 1 ms and provide these values as 32 bit counter values for the CPU. The module is logically divided into 4 blocks of 8 inputs. The inputs of each block can be processed as binary input signals, event or counters, depending on the parameters set.
The input processing (debounce time, edge recognition and inversion) can be configured separately for each input.
The module supports DCF77 formatted time receivers over a 24VDC input.

## Planning

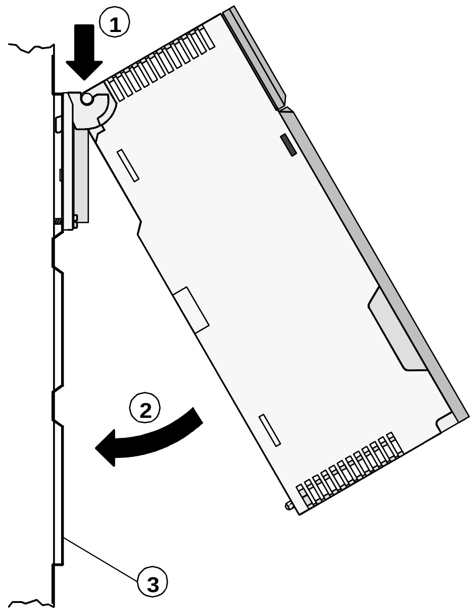
### What is to be planned

You plan:
- a slot in the Quantum rack (local or RIO station).
- the ERT Paramteres. Each of the 4 ERT 854 10 input blocks can be configured with a different functionality (e.g. counters or inputs with our without event recording).
- the connection of the reference voltage for each input group.
- the Process Peripherials Connection.
- the connection of an external time receiver.

### Mounting Position in the Rack

Insert the module in any I/O slot on the TSX Quantum and screw it to the rack. The module must be screwed into position to ensure correct operation (EMC).

Mounting the Module



**1** Insert the module

**2** Screw the module to the rack

**3** Rack

## Module Cabling

**Overview**

This section describes the connection of time receivers, supply voltages and external input signals.

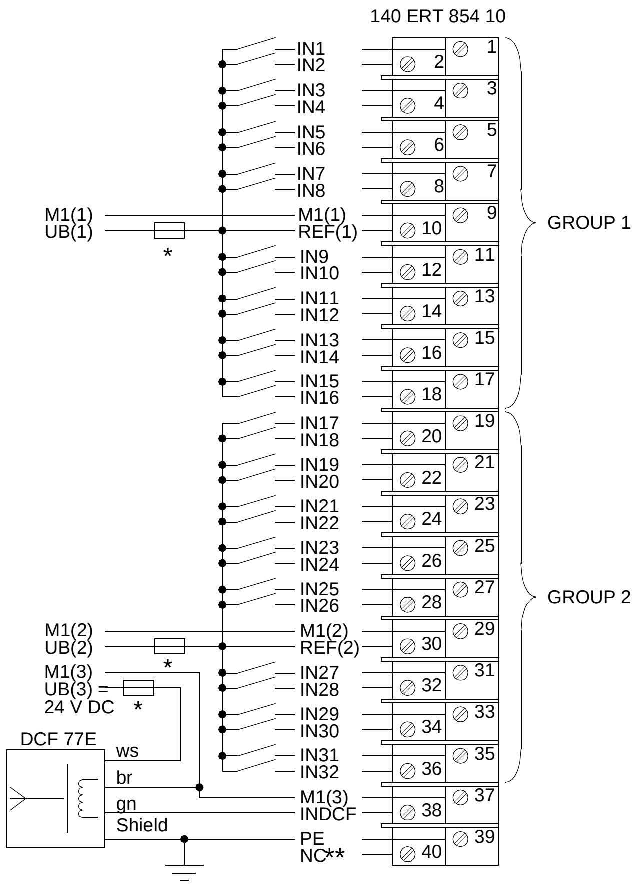
**Reference Voltage**

The input voltage range for the inputs is defined with the reference voltage. Reference voltages and input signals of the same group are to be protected with a common fuse. In addition, the inputs can also be individually protected.

| | **CAUTION** |
|---|---|
| ⚠ | **Damage of the Module**<br>Never use the ERT module without a proper reference voltage to avoid damage to the module.<br>**Failure to observe this precaution can result in injury or equipment damage.** |

**DCF 77E**

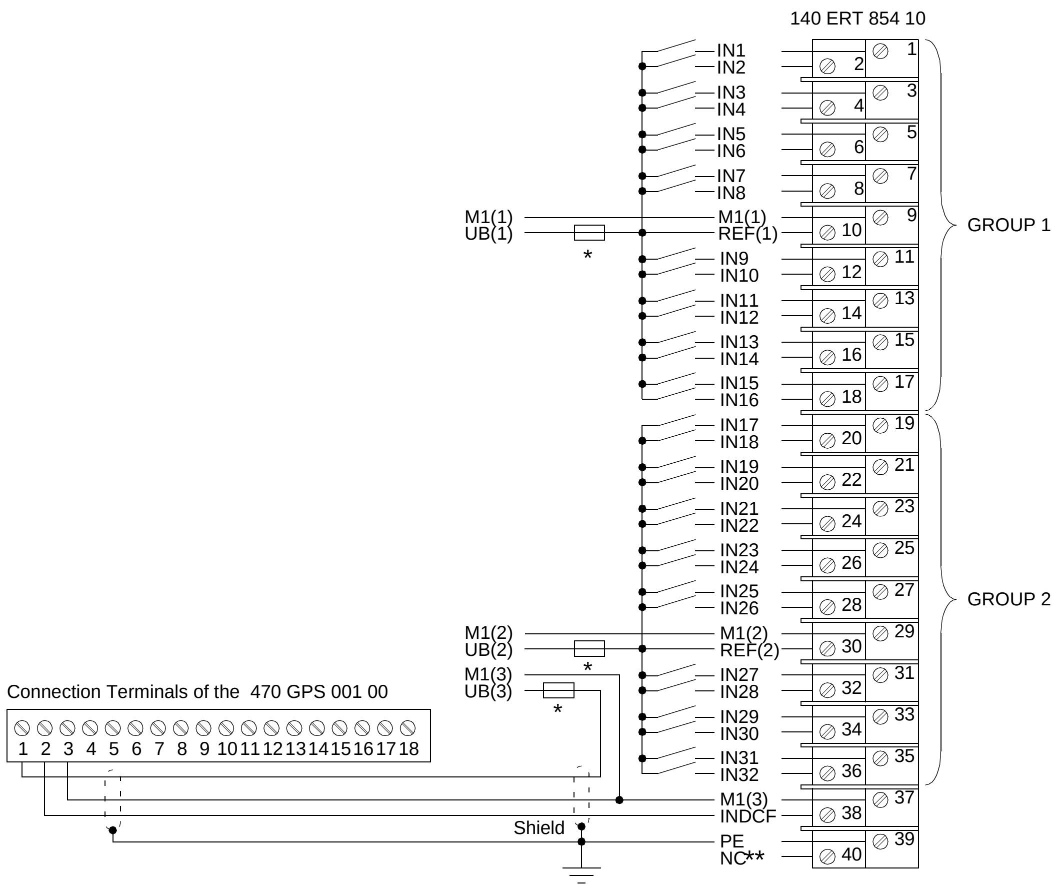
Connection example for the ERT 854 10 with a DCF 77E time receiver



***** UB(1), UB(2):24 ... 125 VDC, UB(3): 24 VDC, separate protection recommended

****** not connected, suitable for support clamp for UB(3)

**GPS 001** Connection example for the ERT 854 10 with a GPS 001 time receiver



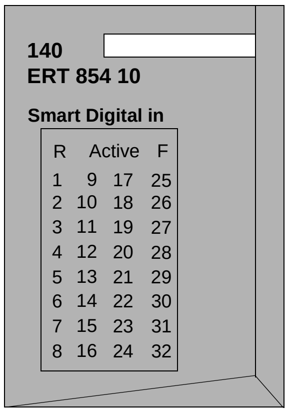
***** UB(1), UB(2):24 ... 125 VDC, UB(3): 24 VDC, separate protection recommended

****** not connected, suitable for support clamp for UB(3)

## Diagnosis

**Condition Display**

The modules have the following indicators:



Meaning of the Indicators:

| Indicators: | Color | Meaning |
|---|---|---|
| R | green | ready. Self test successful when voltage connected The firmware is running correctly and the module is ready for operations. |
| Active | green | The communication with the TSX Quantum CPU is active. |
| F | red | Group Error. Lights when the configured error occurs. |
| 1 ... 32 | green | Input Signal. Indicator for process input signal "1". |

## Technical data

### Supply

Data of the Supply

| | |
|---|---|
| Reference voltage for each process input group | 24 ... 125 VDC, (max. 18 ... 156 VDC)Current consumption per group: max. 3 mA |
| internal via the rack | 5 VDC, max. 300 mA |
| Current requirements for buffer operation | maximum 0.07 mA from XCP 900 00 |

### Process Inputs

Data of the Process Inputs

| | | | | |
|---|---|---|---|---|
| Number | 32 in 2 Groups | | | |
| Input Voltage | 24 ... 125 VDC | | | |
| Potential isolation | Inputs to the Quantum Bus, Group 1 to Group 2 (Opto-coupler) | | | |
| Debounce time | 0 ... 255 Millisecunds (configurable) | | | |
| Inversion | Set with parameters | | | |
| Max. Cable length | 400 m unshielded, 600m shielded | | | |
| Switching Level:<br>Nominal voltage for the input signals<br>Min current for a 1 signal | 24V<br>6mA | 48V<br>2.5mA | 60V<br>2.5mA | 125V<br>1mA |
| Signal level 0 signal<br><br>Signal level 1 signal | nominal 0% of the group reference voltage, max. +15 %, min. -5 %<br>nominal 100% of the group reference voltage, max. 125 %, min. 75 % | | | |
| Internal power loss from all process inputs | max 7.5 W | | | |

**Input for the time receiver**

Data for the time receiver

| | |
|---|---|
| Number | 1, DCF77 Data format from DCF- 077E or GPS - 470 001 00 |
| Input Voltage | 24 VDC |
| Potential isolation | Optocoupler |
| Time Stamp resolution | 1 ms |
| Current consumption | 5 mA |

**Mechanical structure**

Dimensions and Weight

| | |
|---|---|
| Format | Width = 40.34 mm (Standard Housing) |
| Mass (weight) | 0.45 kg |

**Connection Type**

Data of the Connections

| | |
|---|---|
| Process Inputs, DCF receiver | 40 pins Terminal Block |

**Environmental conditions**

Data of the Environmental Conditions

| | |
|---|---|
| System Data | See Quantum User Manual |
| Power loss | Max. 9W, typical 5W |

# Configuration

III

## Introduction

**Overview**

The 140 ERT 854 10 in included as a standard module from version 2.2 of Concept. In this section, the configuration of the modules and the parameter setting of the corresponding EFBs are described. An example is given for the most important applications.

**What's in this part?**

This Part contains the following Chapters:

# The Parameter Screen

6

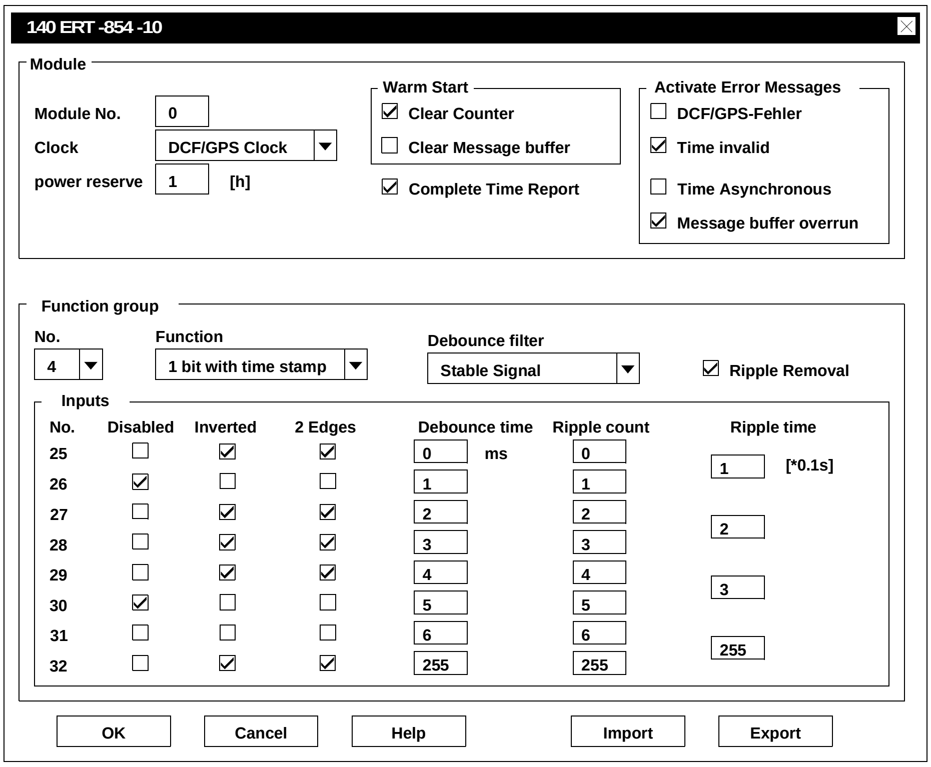
## Parameter Screen

**Call**

To get the parameter screen for the 140 ERT 854 10 module,  select the screen for the I/O configuration of the module and click on the **Params** button.

**Parameter screen layout**

The parameter screen contains general module parameters on four pages which contain specific parameters for each function group. The parameters are preset in the "I/O connection" with default values and can be edited by the user. The export function saves the module parameter settings in a (*.ert) file. The values can be reloaded using the Import function. (These files can also be used as an interface to an Offline parameter tool). Editing parameters is only possible as long as the application program is not running.
Parameter Screen construction


140 ERT -854 -10

Module

| | |
|---|---|
| Module No. | 0 |
| Clock | DCF/GPS Clock |
| power reserve | 1 [h] |

Warm Start
- ✓ Clear Counter
- ☐ Clear Message buffer

✓ Complete Time Report

Activate Error Messages
- ☐ DCF/GPS-Fehler
- ✓ Time invalid
- ☐ Time Asynchronous
- ✓ Message buffer overrun

Function group

| No. | Function | Debounce filter | |
|---|---|---|---|
| 4 | 1 bit with time stamp | Stable Signal | ✓ Ripple Removal |

Inputs

| No. | Disabled | Inverted | 2 Edges | Debounce time [ms] | Ripple count | Ripple time [*0.1s] |
|---|---|---|---|---|---|---|
| 25 | ☐ | ✓ | ✓ | 0 | 0 | 1 |
| 26 | ✓ | ☐ | ☐ | 1 | 1 | |
| 27 | ☐ | ✓ | ✓ | 2 | 2 | 2 |
| 28 | ☐ | ✓ | ✓ | 3 | 3 | |
| 29 | ☐ | ✓ | ✓ | 4 | 4 | 3 |
| 30 | ✓ | ☐ | ☐ | 5 | 5 | |
| 31 | ☐ | ☐ | ☐ | 6 | 6 | 255 |
| 32 | ☐ | ✓ | ✓ | 255 | 255 | |

**Parameter and Default values**

The following list gives an overview of the parameters available and their default values.

**Module**

The following parameters are valid for the entire module.

| Name | Default value | Area | Meaning |
|---|---|---|---|
| Module No. | 0 | 1... 127 | User defined, inserted in event message. The uniqueness of the value is not checked. 0 = Default, no selection made |
| Clock | DCF/GPS clock | DCF/GPS clock | External synchronization in DCF77 format by the DCF or GPS clock. |
| | | Internal clock | Synchronization by telegram; the clock runs either without monitoring or is monitored within a power reserve. |
| | | No | Internal clock is deactivated |
| Power reserve | 1 hour | 1 ... 254 hours | Internal clock: Time from the last synchronization until setting the TU bits and the time until th etime stamp becomes invalid. |
| | | 0 | Internal clock: 0 = free run mode without elapsed time (TE/TU bits are not set) |
| | | 1 ... 5 hours | DCF/GPS clock: 1 hour recommended |
| Complete time report Output | y | n/y | starts/stops the transfer of the complete time telegram (with month and year). Transfer of the complete time report is made as a dummy event 1x directly before a time stamp event: the prerequisite is ALWAYS transferring a time stamp event for monthly transitions, every start/stop of application programs, clearing the time stamp buffer, starting/setting the clock, otherwise the complete time telegram is not sent. |
| Warmstart: | | | |
| Clear counter | n | n/y | Clear counter on Warmstart |
| Clear message buffer | n | n/y | Clear FIFO buffer on Warmstart |
| Activate error message | | | |
| DCF/GPS error | n | n/y | Error values shown by the error LED "F": The enabled bits are treated as errors. Every disabled bit is treated as a warning. (The error bits for an error during a self test are always set). |
| Time invalid: | y | n/y | |
| Asynchronous time | n | n/y | |
| Message buffer overflow | y | n/y | |

**Function group** The following parameters are valid for individual groups (i.e. 4 individual masks)

| Name | Default value | Area | Meaning |
|---|---|---|---|
| No. | 1 | 1....4 | Number of the selected function groups |
| Function | 1 bit with time stamp | Binary | Only binary inputs |
| | | Counter | Binary and counter values |
| | | 1 bit with time stamp | Binary + 1 bit event logging |
| | | 2 bit with time stamp | Binary + 2 bit event logging |
| | | 8 bit with time stamp | Binary + 8 bit event logging |
| Debounce filter | Stable signal | Integrated Stable Signal | Debounce filter mode |
| Dechattering | n | n/y | Disabling/enabling the chatter filter |

**Inputs** The following parameters refer to all individual inputs (attention: chatter time refers to two inputs next to each other)

| Name | Default value | Area | Meaning |
|---|---|---|---|
| No. | 1 - 8 | 1 - 8, 9 - 16, 17 - 24, 25 - 32 | Input number sequence for the function group selected |
| Disabled | n | n/y | Impedes processing of input data for the input (always 0) |
| Inverted | n | n/y | Reverse polarity of the input |
| 2 edges | y | n/y | Edge monitoring for both edges |
| Debounce time | 1 | 0 .. 255 | Debounce time 0 ... 255 milliseconds<br>0 = without internal SW delay |
| Chatter number | 0 | 0 .. 255 | Chatter number 0 ... 255 (for event/counter inputs)<br>0 = chatter filter deactivated |
| Chatter time | 1 | 1 .. 255 | Chatter filter time duration 1 ... 255*0.1 seconds (attention: affects two inputs positioned next to each other!) |

# Startup the140 ERT 854 10

7

## Introduction

**Overview**

This chapter describes the preconditions and boundary conditions required for starting the 140 ERT 854 10 and provides a check list with the necessary steps.

**What's in this Chapter?**

This Chapter contains the following Maps:

## 140 ERT 854 10 Module and Ressource Limitations

**Limitations**

Check whether the following conditions have been adhered to before starting the configuration:

- Concept V 2.2 or higher
- Can be used in local or remote module racks (RIO) with RIO Drop Firmware higher than V1
- Cannot be used in DIO Drops
- Up to 9 ERTs can be mounted on each local or remote module rack (several module racks possible)
- Processing signal status > 1 millisecond + filter time possible
- Counter inputs up to 500Hz with 32 bit addition
- Each ERT requires an "ERT_854_10" transfer EFB
- 7 INPUT words, 5 OUTPUT words per ERT
- Several ERT modules can be connected to one standard time receiver. The 140 ERT 854 10 requires 5 mA from the receiver
- Maximum power consumption of 0.07mA from the battery module XCP 900 00 required for receiving counter, event FIFO buffer and parameter data.

**Time receiver**

The standard time receiver must provide an output signal in DCF77 format for 24 VDC.
The following standard time receivers are provided:

- DCF77E: DCF long wave receiver for Europe
- 470 GPS 001 00: A GPS satellite receiver

## DCF Receiver

**Overview**

The DCF 77E module operates as an internal receiver with integrated antenne. The module receives and converts the received time signal in a 24 VDC signal in DCF77 format, and amplifies it before sending it on to the 140 ERT 854 10 module.

**DCF Signal**

The time signal received in the Central European Time zone is known as the DCF77 and provides CET.   It is sent from the atomic clock to the National Institute for Science and Technology Braunschweig, Germany, and sends a long wave signal of 77.5 kHz (from which DCF77 derives its name) via a transmitter in Frankfurt am Main. The signal can be received throughout Europe (in a radius of approximately 1000 km from Frankfurt).
When selecting a location for erecting an antenne, the following sources of interference should be taken into account which could disturb or destroy signal reception through their DCF receivers:

- electromagnetically contaminated areas. Avoid areas with potential sources of interference, such as strong transmitters, switching stations and airports.  Strong interference can also be caused industrial machinery and cranes.
- Steel supports in buildings, rooms and appartments. Poor reception can occir in cellars, underground car parks and closed operating cabinets.
- "Shadows" and "dead band" in mountain areas, high buildings, ...

## The GPS Receiver

**Overview**

The 470 GPS 001 00 module is a GPS time signal receiver. Other usual GPS standard time receivers can also be used as long as they deliver the time signal in DCF77 format with a 24 VDC potential.

**GPS Signal**

A group of lower orbiting GPS satellites (Global Positioning System) send radio signals from which entensive time information can be derived. Their orbits are distributed evenly so that every point on earth is covered by at least 3 different satellites. The GPS signal can be received accross the whole world. The absolute time precision achieved by the GPS signal is considerably higher than that reached by the DCF receiver.
GPS satellites sends UTC time (Universal Time Coordinated) which corresponds to GMT (Greenwich Mean Time). Seconds and years transitions are taken into account. The 470 GPS 001 can be configured using a time offset from UTC corresponding to the local time zone. Summer/winter time change overs can be configured likewise.
Calendar and day data is diverted from the GPS signal and transferred to the 140 ERT 854 10 module.
The antenne must be ordered separately from the GPS receiver. More details are contained in the technical data section of your reciever.
When selecting a location for erecting an antenne, the following sources of interference should be taken into account which could disturb or destroy signal reception through their GPS receivers:

- electromagnetically contaminated areas: Avoid areas with potential sources of interference, such as strong transmitters, switching stations and airports.
- limitred to the sky and the horizon: The antenne must be erected outside to ensure disturbance operation. Enclosed spaces or operating cabinets impedes satellite reception.
- Length of the antenne cable: Do not exceed the maximum permitted length of the antenne cable
- Atmospheric conditions: Heavy snowfall and rain can impede your GPS receiver or even prevent any signal reception.

## Behaviour when starting/restarting and the data storage

**Cold Start**

This is the default behavior of the ERT when connecting or reconnecting a stabile power supply.

- All recorded events, counter values and the current parameters of the ERT are initialized with a defined state.
- The recording of the process data is delayed until the PLC has been started and can therefore provide the ERT with a valid parameter set.
- Since the ERT does not have a hardware clock, the internal software clock is invalid until it has been synchronized in a suitable form:
  - Depending on the source which has been configured for time synchronization, the time stamps for all recorded events are set to invalid time until either: the internal clock is set with a DPM_Time value using the EFB or time synchronization with an external time signal has occurred.
  - A special case: If the "clock" parameter of the ERT was configured as an "internal clock" in free running mode (with a power reserve of zero), the internal clock starts with a default setting at hour 0 on 1/1/1990.
- If a "complete time report" has been configured, a complete time transfer is done directly before the first recorded event so that the clock synchronization follows.

**Data Storage**

The current data of the ERT 854 10 can be protected from a power loss if the rack has a 140 XCP 900 00 battery module. If the supply voltage falls below a defined limit, it will be recognized by the rack. All recorded data, counter values and the current parameter set are saved in a non-volatile RAM by the firmware and remain until the next warm start (see below). In situations where the saving in the ERT does not happen (5 VDC short circuit or hot swap of the ERT module), a cold start is performed.

**Warm Start**

Reconnecting a stabile supply voltage causes a warm start of the ERT module, as long as the module is in a state where it can store the current data in a consistent form.

- All recorded events, counter values and the current parameters of the ERT are restored from the non-volatile RAM.
- If the "warm start" parameters ("Clear counter"/"clear message buffer") are configured, the recorded events and/or counter values are erased.
- Recording of the process data with the ERT is immediately continued with the same parameter set even if the PLC is not started yet or the remote connection could not be restored at this time.
- Since the ERT does not have a hardware clock, the software clock is invalid until it has been synchronized in a suitable form:
  - Depending on the source which has been configured for time synchronization, the time stamps for all recorded events are set to invalid time until either: the internal clock is set with a DPM_Time value using the EFB or time synchronization with an external time signal has occurred.
  - A special case: If the "clock" parameter of the ERT was configured as an "internal clock" in free running mode (with a power reserve of zero), the internal clock starts with a default setting at hour 0 on 1/1/1990.
- If a "complete time report" has been configured, a complete time transfer is done directly before the first recorded event so that the clock synchronization follows.
- If the corresponding "ERT_854_10" transfer EFB is active in the PLC again, the transfer of the events and counter values in the FIFO buffer of the ERT is continued. Current binary input values and status words are also transferred.
- If the PLC provides a new parameter set when starting which would mean a change in the time of process data evaluation, all recorded events and counter values are cleared since they would no longer be consistent with the new parameter set.

## Check List

### Step by Step

The following steps are to be performed for a successful commissioning of the 140 ERT 854 10:

| Step | Action |
|---|---|
| 1 | Plug the 140 ERT 854 10 module into the local or remote rack. |
| 2 | Connect the designated process peripherals and the standard time receiver to the module(see *Module Cabling, p. 34*). |
| 3 | Do not forget to connect the reference supply voltage for the ERT input groups.<br>**Note:** Please ensure that the notes for installation of antennas for the standard time receiver are followed. |
| 4 | Enter the 140 ERT 854 10 in the I/O map.<br>**Note:** Take special note that the module requires seven 3x registers and 5 4x registers in the signal memory. |
| 5 | Configure the 140 ERT 854 10 in the corresponding parameter screens to provide the required functionality (see *Parameter Screen, p. 43*). |
| 6 | Use the correct EFB from the ANA_IO function block library to provide the "slot" input parameter for the "ERT_854_10" transfer EFB. Either QUANTUM for local or QUANTUM and DROP for remote racks (see *DROP: COnfiguring an I/O Station Rack, p. 63* or *QUANTUM: Configuring a Central Rack, p. 66*). |
| 7 | Define EFB user data structures for the required data types. Events can be "used", for example, by outputting them at a printer or storing them in central data storage. |
| 8 | Use the "ERT_854_10" transfer EFB from the EXPERTS function block library to transfer ERT data (see *ERT_854_10: Data transfer EFB, p. 69*).<br>**Note:** The transfer of new events with the "ERT_854_10" EFB overwrites the previous event information. The user confirmation should be provided only when the data are completely evaluated and no longer needed. |
| 9 | Please note the difference in the behavior of the ERT when starting/restarting depending on if the rack has an XCP module (see *Behaviour when starting/ restarting and the data storage, p. 51*). |

# Integration in the Application Program

# 8

## Introduction

**Overview**

The chapter contains information about how the 140 ERT 854 10 module and respective EFBs are inserted in the Concept application program.

**What's in this Chapter?**

This Chapter contains the following Maps:

## Linking intelligent I/O modules

**Introduction**

To link intelligent I/O modules there are EFBs. The EFBs are arranged so that the FDB program can be designed, almost independent of the hardware module used The project specific information is processed and stored in data structures on the PLC using hardware dependent EFBs (e.g. ERT_854_10). The ERT_854_10 data transfer EFB works with these data structures, which reads the raw values from the Input words (3x), processes them and writes them into the output words (4x) along with the ERT handshake and clock synchronization data. The result of this is that the changes of direct addresses or changes in the input or output parameters are automatically evaluated by the EFBs.

**Division into sections**

Since the evaluation of the configured data is only done once after loading, it is recommended that the EFBs for linking to intelligent modules are divided into several sections.

A division into at lease two sections is recommended.
- Configurations Section
- Processing Section

By division into a configuration section and several processing sections, the CPU load can be reduced because the configuration part (configuration section) only has to be executed once (after a restart or a warm start). The processing section must usually be continually executed.
The configuration section is controlled with the EN input of the corresponding EFB. The EFBs are enabled with internal variables that are set to 1 in the first cycle.

## Configuration Section

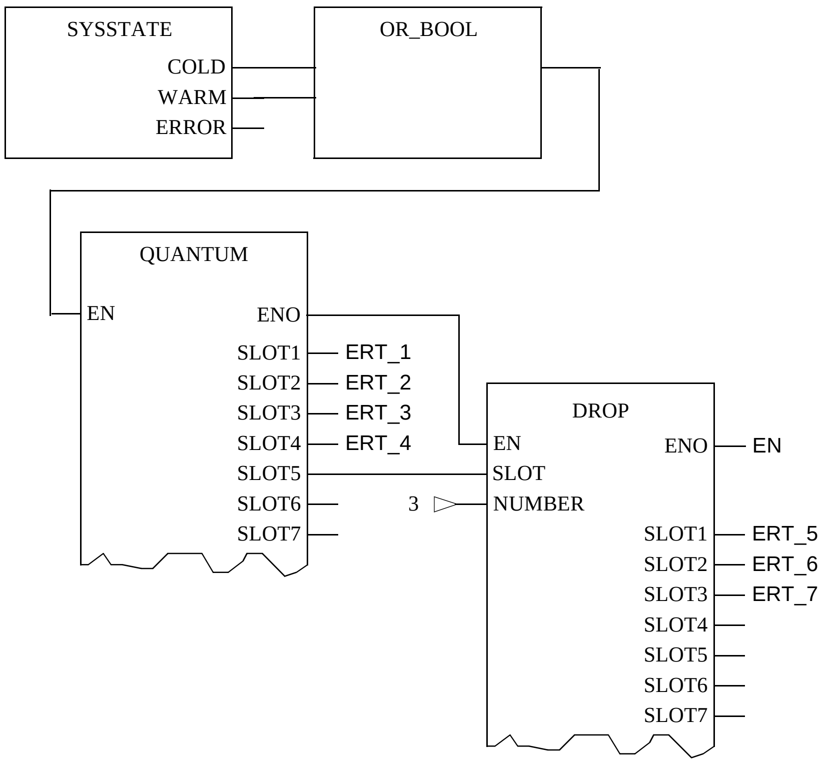
**Configurations Section**

The configuration section is to configure the analog input and output modules and controls the data exchange between the analog EFBs, the Signal memory and the configuration data.
The configuration section should be called "CfgErt" and the internal variable which controls it should be called "CfgErtDone" to guarantee the compatibility to future Concept versions.

There are 2 possibilities for the control of the configuration sections:
- using the EN input of the individual EFBs
- using the enable or disable of the configuration section

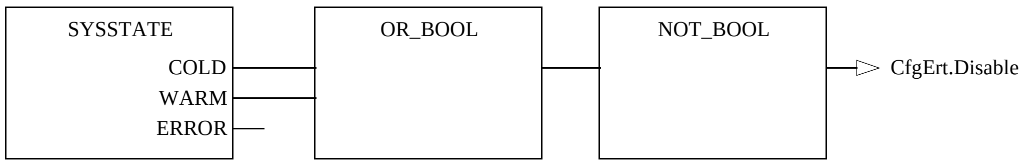
**Example 1: Control using the EN inputs**

The control of the configuration section can be done with the En inputs of the individual EFBs in this section. The enable for the EFBs is done using the EFB SYSSTATE, whose outputs COLD or WARM are set to 1 for a cycle after a cold start or a warm start.
Example of a Configuration Section "CfgErt"



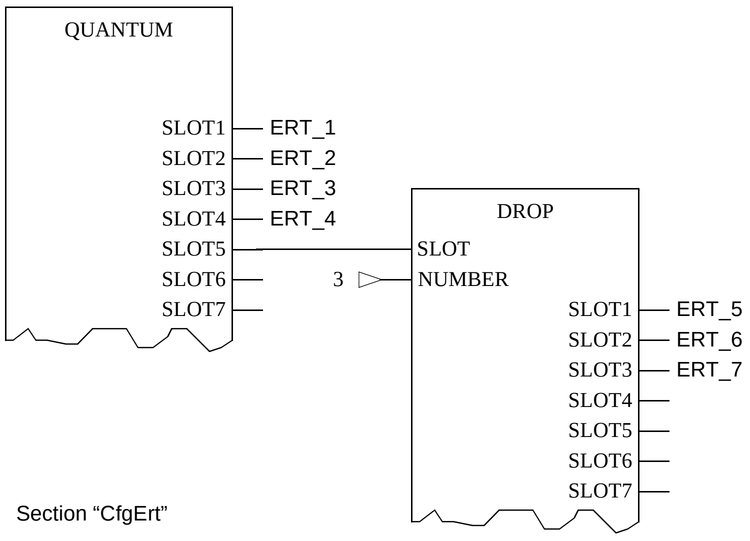
**Example 2: Control using Section Enable**

The control of the configuration section can be done with the enabling and disabling of this section. The enable for the Configuration Section is done in a separate section using the EFB SYSSTATE, whose outputs COLD or WARM are set to 1 for a cycle after a cold start or a warm start. This 1 signal is used to enable and disable the configuration section, A link from EN and ENO of the EFB is not necessary with this solution.
Example of a Control Section "Config_Ctrl"



Section “Config_Ctrl”

Example of a Configuration Section "CfgErt"



Section “CfgErt”

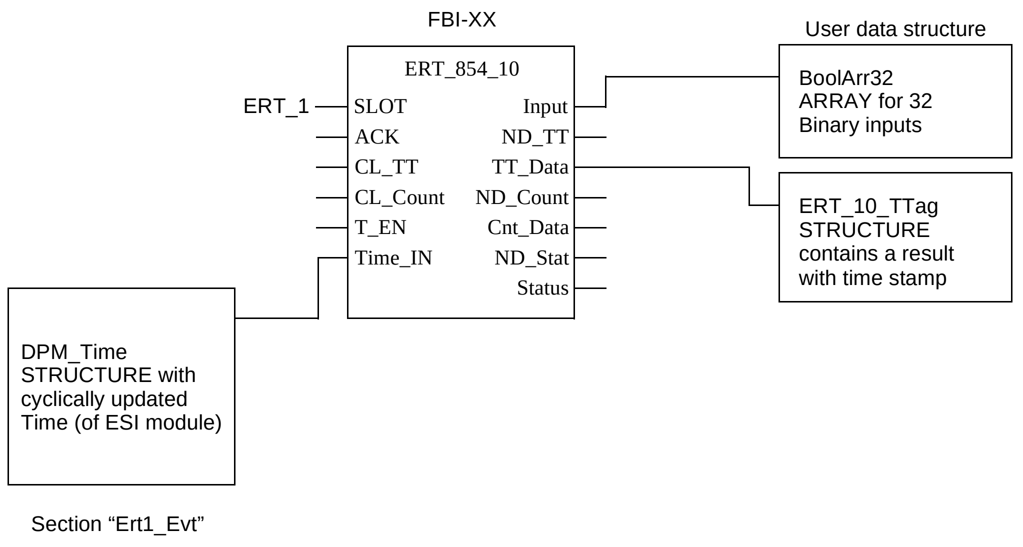
## Processing Section

**Processing Section**

The processing section is for the actual analog value processing.

**Example**

The following example of a processing section uses the parameter "slot" for its ERT_854_10 EFB which can be taken from a QUANTUM or a DROP EFB. (See also *Configurations Section, p. 56*.)
Typical implementation of an ERT_854_10 EFB in the processing section


FBI-XX
ERT_854_10
ERT_1
SLOT
ACK
CL_TT
CL_Count
T_EN
Time_IN
Input
ND_TT
TT_Data
ND_Count
Cnt_Data
ND_Stat
Status
User data structure
BoolArr32 ARRAY for 32 Binary inputs
ERT_10_TTag STRUCTURE contains a result with time stamp
DPM_Time STRUCTURE with cyclically updated Time (of ESI module)
Section “Ert1_Evt”

# EFBs for the140 ERT 854 10

9

## Introduction

**Overview**

The EFBs described in this chapter are required for operating the 140 ERT 854 10.

**What's in this Chapter?**

This Chapter contains the following Sections:

## 9.1 DROP: COnfiguring an I/O Station Rack

### Overview

**Introduction**

This chapter describes the DROP function block.

**What’s in this Section?**

This Section contains the following Maps:

## Short description

**Function description**

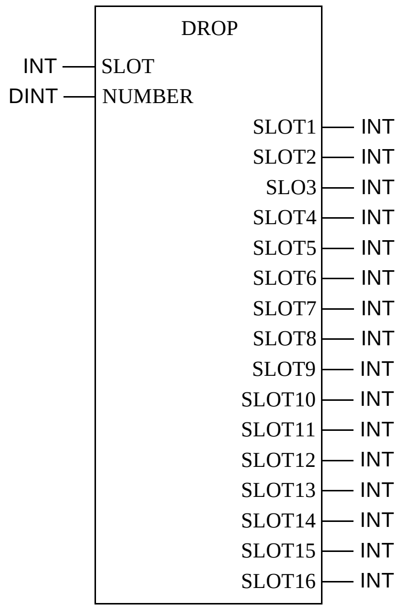
The Function Block is to prepare configuration data of a remote or distributed I/O station for  further processing by module configuration EFBs.
FOr configuration an I/O station rack, the function block DROP in the Configuration Section is connected to the corresponding SLOT output of the function block QUANTUM. At the NUMBER input of the function block DROP, the number of the I/O station must be given, as defined in the I/O connections. The SLOT output connects to the function block for configuration of the analog modules of the I/O station.
The parameters EN and ENO can used additionally.

**Note:** The module ERT 854 10 can not be used in the distributed I/O stations (DIO).

## Representation

### Symbol

Block representation:



### Parameter description

Block parameter description:

| Parameter | Data type | Meaning |
|---|---|---|
| SLOT | INT | Slot for RIO, DIO, NOM |
| NUMBER | DINT | Number of RIO, DIO, NOM |
| SLOT1 | INT | Slot 1 |
| : | : | : |
| SLOT16 | INT | Slot 16 |

## Runtime error

### Runtime error

If no "Head" has been configured for the I/O station subrack, an error message appears.

## 9.2 QUANTUM: Configuring a Central Rack

### Overview

**Introduction**

This chapter describes the QUANTUM function block.

**What's in this Section?**

This Section contains the following Maps:

## Brief description

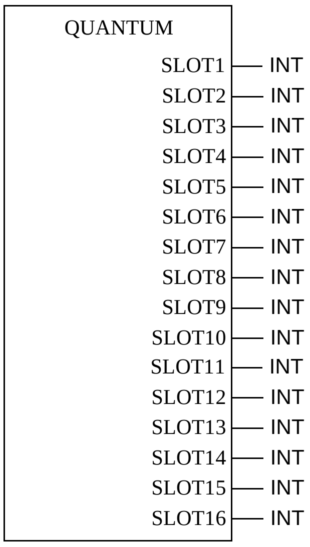
**Function description**

The function block is used to edit the configuration data of a QUANTUM primary subrack for subsequent use by the scaling EFBs.
To configure a QUANTUM primary subrack, the QUANTUM Function block is inserted into the configuration section. The function blocks for the configuration of analog modules or the DROP Function block for the I/O station are connected at its SLOT outputs.
EN and ENO can be projected as additional parameters.

## Representation

**Symbol**

Block representation:



**Parameter description**

Block parameter description:

| Parameter | Data type | Meaning |
|---|---|---|
| SLOT1 | INT | Slot 1 |
| : | : | : |
| SLOT16 | INT | Slot 16 |

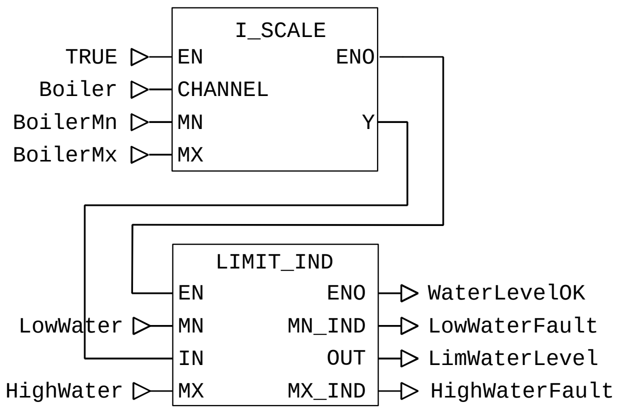
## Application example for Quantum

**At a Glance**

To precisely monitor the output values, it is advisable to implement the scaling with two EFBs. The first EFB (scaling EFBs) scales the analog value and the second EFB monitors the scaled value for ranges preset by the process. In the following process, either the original Y output of the scaling EFB or the limited OUT output of the Limiter EFB can be used.

**Application example**

A simple example shows how the EFBs can be used.
The example assumes a boiler with a capacity of 350 liters. The input voltage ranges from 0.0 Volt for 0 liters to 10.0 Volt for 1000 liters. A PI controller should guarantee a volume between 200 and 300 liters. The Limiter EFB detects violations in this range and will limit the output.
Given values:
BoilerMn: 0
BoilerMx: 1 000
LowWater: 199
HighWater: 301
"Boiler" is an unlocated variable of the ANL_IN type and is linked to an AVI030-EFB.
Application example



## 9.3 ERT_854_10: Data transfer EFB

### Overview

**Introduction**

This chapter describes the ERT_854_10 module.

**What's in this Section?**

This Section contains the following Maps:

## Brief description

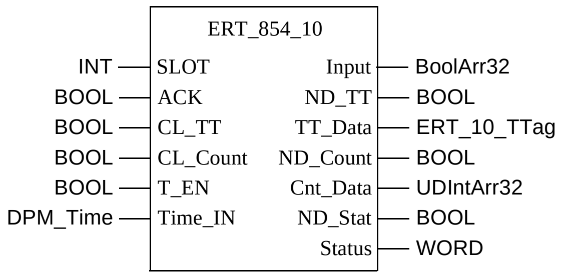
**Function description**

The ERT_854_10 EFB provides the programmer with a software interface to the ERT 854 10 module. It allows easy access to functions like counters, time stamp, status or time synchronization. Using the input and output registers, the ERT_854_10 EFB can coordinate the flow of Multiplex data from the ERT to the PLC. It also ensures that the intermediate counter values are stored in an internal memory area until the data is complete, so a consistent set of all counter values is made available to the statement list. A flag "New data" is always set for every data type if the input data type was copied into the corresponding EFB output structure. The parameters EN and ENO can also be configured.

## Representation

**Symbol**

Function Block representation:



**Parameter description**

Description of the function block parameters:

| Parameter | Data type | Meaning |
|---|---|---|
| SLOT | INT | The Slot index is assigned to the ERT-EFB from either the QUANTUM EFB or DROP EFB and contains the configured input and output references (3x and 4x registers) |
| ACK | BOOL | Event confirmation: Setting ACK signals that the user is ready to receive the next result and deletes the TT_Data register. If ACK remains set, "continuous operation" is done. |
| CL_TT | BOOL | Delete the ERT event FIFO buffer by setting CL_TT. Storage of events is blocked until the CL_TT is reset to 0. |
| CL_Count | BOOL | Delete all ERT counters by setting CL_Count. Counting is interrupted until CL_Count is reset to 0. |
| T_EN | BOOL | Enables a time transfer, e.g. from the ESI via Time_IN, if set |
| Time_IN | DPM_Time | Structure of the input time, e.g. from the ESI, for time synchronization of the ERT (contains the edge controlled time synchronization in the "Sync" element) |
| Input | BOOLArr32 | Output array for all 32 digital inputs in BOOL format<br>(also provided in the form of word references as 3x registers 1+2) |
| ND_TT | BOOL | Flag, new data in TT_Data structure: remains set until user confirmation with ACK |
| TT_Data | ERT_10_TTag | Event message output structure with time stamp. An event is held and NDTT is set to 1 until there is a user enable with ACK = 1. |
| ND_Count | BOOL | Flag, new counter data in Cnt_Data structure: The value 1 is set for only one cycle and is not acknowledged. |
| Cnt_Data | UDIntArr32 | Output array for 32 counter values (is overwritten after the EFB has received a complete set of consistent counter values (configured as:8, 16, 24, or 32). |
| ND_Stat | BOOL | Flag; new status data in status word: The value 1 is set for only one cycle and is not acknowledged. |
| Status | WORD | Output word for EFB/ERT status (for internal details see *Data Flow, p. 76*) |

**Internal time synchronization**

Structure of DPM_Time for ERT internal time synchronization e.g. via the ESI:

| Element | Element type | Meaning |
|---|---|---|
| Sync | BOOL | Clock synchronization with positive edge (hourly or on command) |
| Ms | WORD | Time in milliseconds |
| Min | BYTE | Time invalid / minutes |
| Hour | BYTE | Summer time / hours |
| Day | BYTE | Day of the week / Day in the month |
| Mon | BYTE | Month |
| Year | BYTE | Year |

**Event structure**

Event structure of the ERT_10_TTag with 5 Byte time stamp (further information can be found in *Data Flow, p. 76*):

| Element | Element type | Meaning |
|---|---|---|
| User | BYTE | Complete time / user number [module number] |
| Input | BYTE | Event set type / No. of the first input |
| In | BYTE | Event data: 1, 2 or 8 managed positions |
| Ms | WORD | Time in milliseconds |
| Min | BYTE | Time invalid / minutes |
| Hour | BYTE | Summer time / hours |
| Day | BYTE | Day of the week / Day of the month |

## Mode of Functioning

**ERT data transfer**

The number of I/O words available on the local and remote subracks is limited to 64 inputs and 64 outputs. For this reason, the number of ERT modules which can be used per local/remote backplane is limited to 9, with the currently selected minimum requirements of 7 input words and 5 output words per module.

The size of the required ERT data transfer is considerably larger:
- 32 counters = 64 words,
- an event with a 5 byte time stamp = 4 words,
- 32 digital values and the ERT status = 3 words.

These inconsistent size requirements necessitate the use of a special transfer EFB called ERT_854_10 to execute the required operations on the PLC and to adjust the ERT representation of the data in Multiplex form. An EFB is required for every ERT module.
To simplify matters, only the EFB parameters which will actually be used need to be configured. This saves on the amount of configuration effort, particularly when the counter inputs and event inputs are not mixed together. Unfortunately memory cannot be reserved for this because Concept has occupied the outputs with invisible dummy variables.
Basic structure of the ERT_854_10 input register block with seven 3x registers for transfer from the ERT to the PLC.

**Basic structure of the register block**

ERT_854_10 input register block:

| Contents | Function |
|---|---|
| Digital inputs 1 …. 16 | Digitally processed input data which is cyclically updated (the module’s input address corresponds to that of the digital standard input modules, i.e. inputs 1 … 16 correspond to bits 15 … 0) |
| Digital inputs 17 …. 32 | |
| Transfer status | IN transfer status (TS_IN) |
| MUX 1 | Multiplex data block for block transfer, such as: |
| MUX 2 | 1 event with 5 byte time stamp or |
| MUX 3 | 2 counter values of  possible configured maximum 32 or |
| MUX 4 | 1 status word |

Simplified structure of the ERT_854_10 output register block with five 4x registers for the transfer of the SPS to the ERT.

ERT_854_10 output register block:

| Contents | Function |
|---|---|
| Transfer status | OUT transfer status (TS_OUT) |
| MUX 1 | Time data block for the ERT for the clock synchronization |
| MUX 2 | |
| MUX 3 | |
| MUX 4 | |

**Note:** User interface is normally for the inputs and outputs of the ERT_854_10 EFB, not the 3x and 4x registers.

## Use of the DPM_Time Structure for the synchronization of the internal ERT clock

**Time synchronization**

If the time can not be synchronized through a standard time receiver, the time information can alternatively be transferred from the 140 ESI 062 01 communication module. The ESI makes the updated time available in a DPM_Time structure directly using the "Time_IN" parameter. The data structure can also be filled by the user program and the corresponding bits can be set. In this manner, the time can also be set, for example, by the CPU.

**With power reserve**

As soon as the "clock" parameter of the ERT is configured to "internal clock" with a power reserve not equal to zero (i.e. not free running), the EFB must use the time provided by the ESI for synchronizing the internal ERT clock. Until the first synchronization has taken place, the ERT sends back "status" output word with the bit "invalid time" set (Bit 3 TU).
The conditions of the first synchronization of the internal ERT using above the DPM_Time structure are:
The EFB Parameter "T_EN" must change from 0 to 1 to enable the time setting.

The time in "TIME_IN" provided by ESI must be represented as follows:
- valid (i.e. the bit for the message "time invalid" in "Min" value must not be set),
- and the values in "Ms" must change continually.

If, at a later point in time, the time data is invalid or no longer set, the TU changes to 1 after the configured power reserve has run out.

The synchronization/setting of the internal ERT clock takes place using the DPM_Time structure, if:

- EFB-Parameter "T_EN" is set to 1 to enable the time setting.
- The time data in "Time_IN" provided by ESI are valid (i.e. the "Time invalid" Bit in the "Min"value must not be set).
- The status of the DPM_Time element "Sync" changes from 0 to 1. This change is done every complete hour by the 140 ESI 062 01, but can also be triggered by a suitable telecontrol command.

The precision of the ESI and ERT synchronized time can be influenced by delay caused by the PLC cycle time, as well as by the cumulative components, which reflect the differences of the ERT software clock (< 360 milliseconds/second).

**Without power reserve**

If the "clock" parameter of the ERT was configured as an "internal clock" in free running mode (with a power reserve of zero), the internal clock starts with a default setting at hour 0 on 1/1/1990. In this case, the time can also be provided by using the DPM_Time data structure of the 140 ESI 062 01 module, as described above. As there is no power reserve to "run out", the time will never be invalid and the bit "Time not synchronized" is always set in the "status" output word (Bit 4  TA) which is returned by the EFB, .

## Data Flow

**Digital Inputs**

No flag for new data is provided for this input type. The digital inputs in the first two input register words are updated every second cycle directly by the ERT. The EFB makes the processed values available as Bool if the BoolArr32 output field has been configured accordingly.

**Counter Inputs**

Cyclic updating of the counter values takes significantly longer than for other data types. Counter values are saved as a data record in "Cnt_Data" after a complete series (configured as: 8, 16, 24 or 32) of time consistent counter values in multiplex form has been transferred from the ERT. The flag for new data "ND_Count" is set for one cycle.

**Event Inputs**

As readiness to receive new events must be actively confirmed by the user, the management of the registers becomes somewhat more complex (a handshake mechanism is required). Event data remain in the data structure ERT_10_TTag and the flag for new data "ND_TT" stays set until the "ACK" input is set by the user and therefore requests a new event. The EFB responds to this by resetting "ND_TT" for at least one cycle. After the new event has been sent to the ERT_10_TT register structure, "ND_TT" is reset by the EFB. To prevent the new event data from being overwritten, the user must take care that the "ACK" input is reset after the EFB has reset the "ND_TT" flag. This state can then be kept stable to allow the user program enough time for event processing. Each subsequent event which is recorded with the ERT is temporarily stored within the event FIFO buffer.
New events are sent directly from the internal buffer of the EFB in intervals of at least 2 cycles for as long as the "ACK" input is set (for the special continuous operating mode); the effect is, however, that the "ND_TT" only stays set for one cycle. In this special mode, it is still the job of the user program to finish event processing before "ND_TT" signals the transfer of other new events to the ERT_10_TT structure because handshake protection by "ACK" is not available in this case.

**ERT_10_TTag**

ERT_10_TTag event structure with 5 byte time stamps

| Byte | Bits | Function |
|---|---|---|
| 1 | D0...D6 = Module No.. 0...127<br>D7 = CT | Rough time: CT = 1 indicates that this time stamp contains the whole time value including month and year in bytes 2 + 3. The Module no. can be set in any way in the parameter screen. |
| 2 | D0…D5 = input no.<br>D6 = P1<br>D7 = P2 | No. of the first input of the event group: 1...32<br>Type of the event message (P2, P1). 1.. 59 see *Note 1:, p. 77*<br>[Month value if CT = 1] |
| 3 | D0…D7 = data from the event group (D7…D0 with right alignment) | 1, 2 or 8 managed positions<br>[Month value if CT = 1] |
| 4 | Time in milliseconds (least significant byte) | 0 ... 59999 milliseconds (max. 61100) see *Note 2:, p. 78* |
| 5 | Time in milliseconds (most significant byte) | 0 ... 59999 milliseconds (max. 61100) see *Note 2:, p. 78* and *Note 3:, p. 78* |
| 6 | D0...D5 = minutes<br>D6 = R<br>D7 = TI | Minutes: 0...59<br>Time invalid: TI = 1 means invalid time / reserved = 0 see *Note 3:, p. 78* |
| 7 | D0...D4 = hours<br>D5 = R<br>D6 = R<br>D7 = DS | Hours: 0...23<br>Summer time: DS = 1 indicates that summer time is set With switchover from ST -> WT, hour 2A has ST, and hour 2B has WT |
| 8 | D0...D4 = DOW<br>D5...D7 = DOM | Weekday: Mon-Sun = 1…7<br>Day of the month: 1...31<br>The code corresponds to CET and thus deviates from the standard used in the US, Sun = 1. |

**Note 1:**

Interpretation for Byte 2

| D7 D6 | Type of the event message | D5...D0 | No. of the first input of the event group |
|---|---|---|---|
| 0 1 | 1 pin message | 1 ... 32 | Input pin number |
| 1 0 | 2 pin message | 1, 3, 5, ...31 | First input of the group |
| 1 1 | 8 pin message | 1, 9, 17, 25 | First input of the group |

**Note 2:**

The value for the milliseconds is a maximum of 61100 ms with the second of transition (61000 plus a tolerance of 100 milliseconds)

**Note 3:**

For time stamps containing an invalid time (TI = 1), the time in milliseconds is set to FFFF HEX. Minutes, hours and DOW/DOM values are invalid (i.e. undefined).

**Rough Time Output**

If the "rough time declaration" has been activated during the ERT configuration, the transfer of the complete time (with month/year) is executed under the following conditions: when the month changes, after the module restarts, during every start or stop of the PLC user program, when the event FIFO buffer is deleted, when the clock is started or set. The transfer of this complete time output without the data input values is "triggered" basically takes place through a correct time stamped event. If this does not happen the values remain "stuck" in the ERT until an event occurs. Within the time stamp of a "rough time output", the CT bit is always set so that byte 2 contains the information about the month, byte 3 the information about the year and bytes 4 to 8 show the same time stamp values of the triggering event, which is immediately followed by the event message for rough time output.

**Status Inputs**

The flag for new status data "ND_Stat" is set for one cycle. The status inputs can be overwritten after 2 query cycles.
The status word contains EFB and ERT error bits.

**Assignments of the Error Bits**

Internal structure of the EFB/ERT status word:

| **EFB error bits** | | | | | **ERT error bits** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D15 ... | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

## ERT Error Bits

D8 ... D0   ERT error bits

| Bit | Abbreviation | Meaning |
|---|---|---|
| D0 | FW | Firmware errors, self test errors within EPROM, RAM or DPM (severe module errors) |
| D1 | FP | Parameter errors (severe internal errors) |
| D2 | TE | External time reference error (time-basis signal disrupted or not available) |
| D3 | TU | Time became invalid |
| D4 | TA | Time is not synchronized (Free running mode, permanent running without time error message, see also:*Without power reserve, p. 75* |
| D5 | PF | FIFO buffer overflow (loss of the most recent event data) |
| D6 | PH | FIFO buffer half full |
| D7 | DC | Dechattering active (some event data lost) |
| D8 | CE | ERT communication error (procedure errors or time out) |

When configuring the Screens (See *Parameter Screen, p. 43*) parameter, some of these errors can be assigned to grouped error messages with the "F" light as well as the module's error byte within the status table.  All other errors are then defined as warnings.
D11 ... D9   reserved

## EFB Error Bits

D15 ... D12   EFB error bits:

| Bin. | Hex. | Meaning |
|---|---|---|
| 1000 | 8 HEX | EFB communication time out |
| 0101 | 5 HEX | Wrong slot |
| 0110 | 6 HEX | Health status bit is not set (ERT appears not to be available) |
| other values | | internal error |

**Online error display**

The following ERT/ERB error messages are displayed in the **Online** → **Event viewer** Concept window with an error number and explanation.
EFB error messages:

| Message | Error | Meaning |
|---|---|---|
| -2710 | User error  11 | EFB communication time out |
| -2711 | [User error 12] = | EFB internal error |
| -2712 | [User error 13] = | EFB internal error |
| -2713 | [User error 14] = | EFB internal error |
| -2714 | [User error 15] = | EFB internal error |
| -2715 | [User error 16] = | Wrong slot |
| -2716 | [User error 17] | Health status bit is not set (ERT appears not to be available) |
| -2717 | [User error 18] | EFB internal error |

ERT error messages:

| Message | Error | Meaning |
|---|---|---|
| -2700 | User error  1] | ERT internal error |
| ... | ... | ... |
| -2707 | [User error 8] | ERT internal error |
| -2704 | [User error 5] | ERT communication timeout (e.g. EFB disabled too long) |

## Other functions

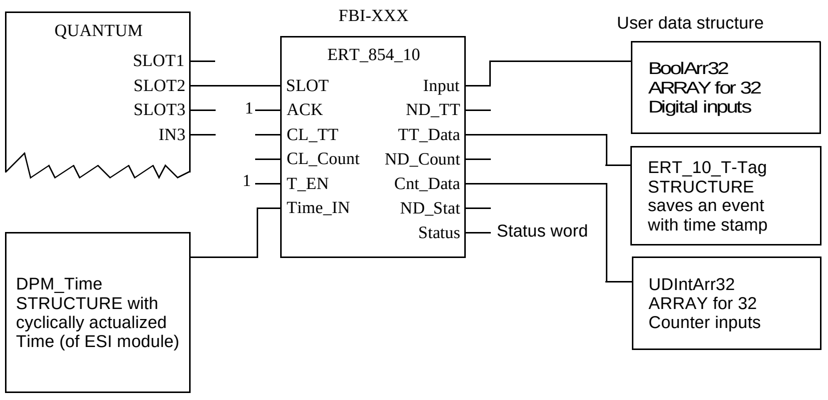
**Input markers**

Setting the input marker "CL_TT" causes the FIFO buffer event of the ERT to be cleared. Setting the markers for one cycle is sufficient.
Setting the input marker "CL_Count" causes the ERT counter to be cleared by the ERT. Setting the markers for one cycle is sufficient.

## Simple example

**Structure diagram**

Principle structure


QUANTUM
SLOT1
SLOT2
SLOT3
IN3
FBI-XXX
ERT_854_10
SLOT
ACK
CL_TT
CL_Count
T_EN
Time_IN
1
1
Input
ND_TT
TT_Data
ND_Count
Cnt_Data
ND_Stat
Status
Status word
User data structure
BoolArr32 ARRAY for 32 Digital inputs
ERT_10_T-Tag STRUCTURE saves an event with time stamp
UDIntArr32 ARRAY for 32 Counter inputs
DPM_Time STRUCTURE with cyclically actualized Time (of ESI module)

## Using the ERT >EFB time data flow

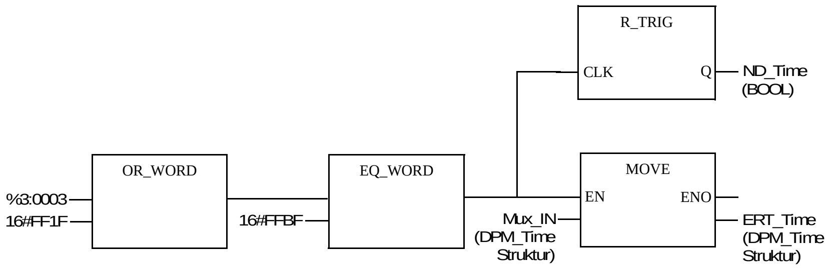
**Application Examples:**

This section shows an internal function which is made available by the ERT for diagnostics and development. It covers the cyclic transfer of the ERT internal time to the corresponding EFB in greater intervals. This time can be used for display or setting the PLC clock and so on, irrespective of whether it comes from the free-running internal clock or was synchronized through an external reference clock signal. The time appears as a DPM_Time structure beginning at word 4 of the IN register block of the ERT. The following diagram shows the program elements involved in selection.

**Startup information:**

During the I/O addressing, the IN references 30001 …30007 were assigned to an ERT_854_10. The IN transfer status (TS_IN) in the third word of the register block is sent to an OR_WORD block. A DPM_Time structure is defined in the variable editor as Variable Mux_IN in the fourth word of the IN register block  and has address 30004 ... 30007. This variable is given as an input to the MOVE block. The MOVE block output is a DPM_Time structure defined by the variable editor as variable ERT_Time.

Typical recording mechanism for ERT time data



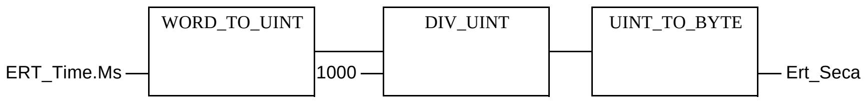
**Note:** The ERT_854_10-EFB must be active and error free.

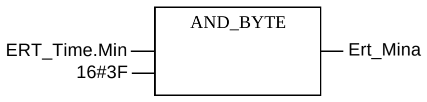
**Explanation:** The MOVE block transfers the time data (which is cyclically stored in the MUX range of the IN register block) to the DPM_Time structure ERT_Time of the user as soon as the OR_WORD and the EQ_WORD block signal for a time data transfer. R_TRIG provides a signal in "ND_Time" for one cycle to allow further processing of the time data. The BOOL "Sync" element value of the ERT_Time should begin to "tick" during each new transfer from the ERT. There is a new transfer after a maximum of each 200 PLC cycles.

**Example 1: Using time values for display (or with SET_TOD-EFB)** A number of simple logical operations is needed to obtain a meaningful display of the time information of the DPM_Time structure. The same commands can also be used for the ERT_10_T Tag structure. As example 2 deals with setting the PLC clock while using the SET_TOD-EFB, individual values are directly converted into the required formats.

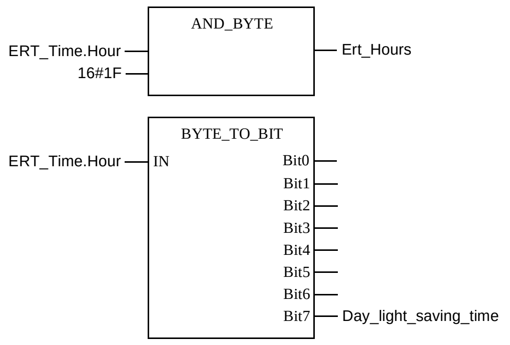
**Note:** The reference data editor (RDE) can provide the "ms" value directly in the Uns-Dec-WORD format and the "Min" value in the Dec-BYTE format.

SET_TOD requires that the WORD millisecond value "ms" is converted into a BYTE second value. The BYTE minute value "Min" contains the error bit which must be removed (values greater than 127 are invalid).
Conversion of the WORD millisecond value into a seconds BYTE

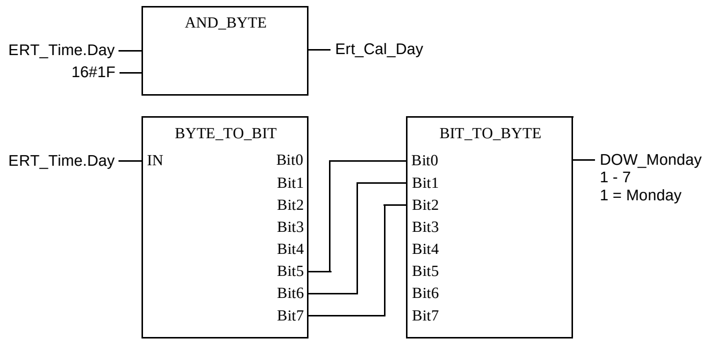




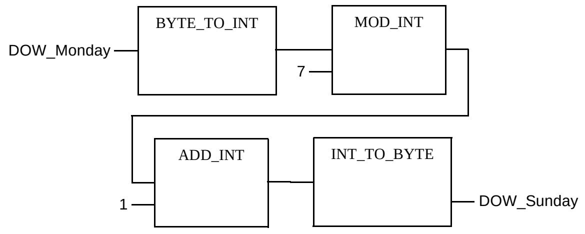
The BYTE value "Day" contains week and calendar day values. The weekday Monday is displayed as 1 in the DPM_Time structure. The weekday parameter in SET_TOD uses the value 1 for Sunday.
Removing/restoring the bit for the summer time of the "Hour" value.



The BYTE value "Day" contains week and calendar day values. The weekday Monday is displayed as 1 in the DPM_Time structure. The weekday parameter in SET_TOD uses the value 1 for Sunday.
Using the calendar day and weekday based on Monday



Further steps must be taken to convert the weekday based on the value of 1 for Monday into the value of 1 for Sunday.
Calculating the remainder values (Mod) and addition for converting the weekday values

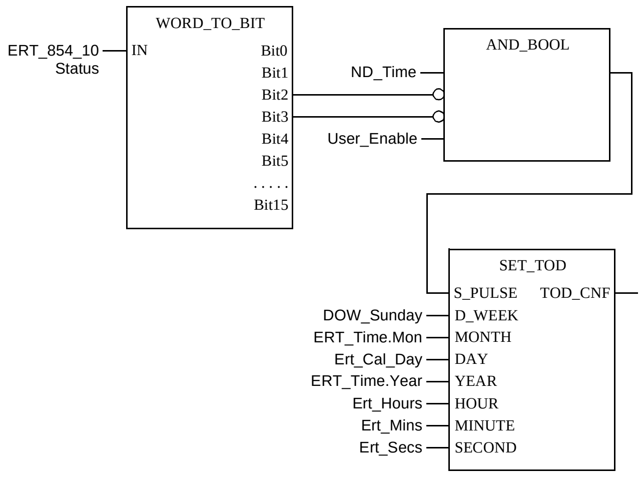


**Example 2: Setting the PLC clock with the SET_TOD EFB while using ERT time data**

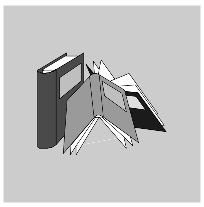
All the parameter values required for the SET_TOD-EFB were created in example 1. The "ND_Time" signal required for transferring the time into the DPM_Time structure with the MOVE block is combined with a user enable here (e.g. only once per hour) to set the PLC clock only when new, error-free time data have been transferred by the ERT. (The ERT error bits are never set when the internal clock is in free run mode).
The SET_TOD-EFB is in the HSBY group of the SYSTEM block library. If it is used, the clock must be activated by storing the TIME OF DAY register in the SPECIALS range of the configuration with 4x addresses.

**Note:** The "status" parameter value is not exactly synchronized with the time data flow and for this reason can only "tend" to reflect the correct value.

User-enabled setting system for the PLC clock while using the SET_TOD-EFB



# Glossary



## A

| | |
|---|---|
| **Action signals** | Action signals represent the states of the action outputs assigned in the application program (e.g. "motor moving" signal). |
| **Active window** | The window that is currently selected. Only one window can be active at a given point in time. If a window becomes active, the colour of its title bar changes to differentiate it from the other windows. Windows not selected are inactive. |
| **Addresses** | (Direct) addresses are areas of memory in the PLC. These are in the signal memory and can be allocated to input/output modules. |
| **Application window** | The window that contains the workarea, the menu bar and the tool bar for the application program. The name of the application program appears in the title bar. An application window can contain several document windows.<br>In the Handtableau, the application window corresponds to a project. |
| **Atrium** | The PC-based controller is fitted to a standard AT circuit board and can be operated in a host computer in an ISA bus slot. The module has a motherboard (requires SA85 driver) with two slots for PC104 daughter boards. Of these, one PC104 daughter board is used for the CPU and the other for the control of the INTERBUS. |

## B

| | |
|---|---|
| **Base 16 literals** | Base 16 literals are used to represent integer values in the hexadecimal system. The base must be identified by means of the prefix 16#. The values are not permitted to have a sign (+/-). Individual underscore characters ( _ ) between the digits are not significant.<br><br>Example<br>16#F_F or 16#FF (decimal 255)<br>16#E_0 or 16#E0 (decimal 224) |
| **Base 2 literals** | Base 2 literals are used to represent integer values in the binary system. The base must be identified by means of the prefix 2#. The values are not permitted to have a sign (+/-). Individual underscore characters ( _ ) between the digits are not significant.<br><br>Example<br>2#1111_1111 or 2#11111111 (decimal 255)<br>2#1110_0000 or 2#11100000 (decimal 224) |
| **Base 8 literals** | Base 8 literals are used to represent integer values in the octal system. The base must be identified by means of the prefix 8#. The values are not permitted to have a sign (+/-). Individual underscore characters ( _ ) between the digits are not significant.<br><br>Example<br>8#3_77 or 8#377 (decimal 255)<br>8#34_0 or 8#340 (decimal 224) |
| **Basic functions/ function blocks (EFB)** | Term for functions or function blocks for which the type definitions are not defined in one of the IEC languages, i.e. their bodies, e.g., cannot be modified using the DFB editor (Concept-DFB). EFB types are programmed in "C" and are provided in pre-compiled form in libraries. |
| **BOOL** | BOOL stands for the data type "boolean". The length of the data elements is 1 bit (stored in 1 byte in the memory). The range of values for this type of data is 0 (FALSE) and 1 (TRUE). |
| **BYTE** | BYTE stands for the data type "sequence of 8 bits". The entry can be a base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 8 bits. A range of numerical values cannot be assigned to this data type. |

## C

| | |
|---|---|
| **Call** | The process that is initiated by the execution of an operation. |
| **Constants** | Constants are unlocated variables to which a value is assigned that cannot be changed by the program logic (write-protected). |
| **Current parameter** | Currently connected input-/output parameter. |

## D

**Data types**

The overview shows the hierarchy of the data types, as they are used for inputs and outputs for functions and function blocks. Generic data types are identified by the prefix "ANY".

- ANY_ELEM
  - ANY_NUM
    ANY_REAL (REAL)
    ANY_INT (DINT, INT, UDINT, UINT)
  - ANY_BIT (BOOL, BYTE, WORD)
  - TIME
- System data types (IEC extensions)
- Derived (from ’ANY’ data types)

**Defined literals**

If you want to define the data type for a literal yourself, you can do this using the following construction: ’data type name’#’value of the literal’.

Example
INT#15 (data type: integer, value: 15),
BYTE#00001111 (data type: byte, value: 00001111)
REAL#23 (data type: real, value: 23)

For the allocation of the REAL data type, it is also possible to enter the value in the following way: 23.0.
If a decimal place is entered, the REAL data type is automatically assigned.

**Derived data type**

Derived data types are data types that have been derived from the basic data types and/or other derived data types. Derived data types are defined in the Concept data type editor. A differentiation is made between global data types and local data types.

| | |
|---|---|
| **Derived Function Block (DFB)** | A derived function block represents the call for a derived function block type. You will find details on the graphic form of the call in the definition "function block (instance)". Contrary to calls for EFB types, calls for DFB types are marked with double vertical lines on the left and right sides of the square block symbol.<br>The body of a derived function block type is written in FBD language, however only in the current version of the programming system. Other IEC languages can currently not be used for the definition of DFB types, derived functions can also not be defined yet in the current version.<br>A differentiation is made between local and global DFBs. |
| **DINT** | DINT stands for the data type "double integer". The entry can be an integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 32 bits. The range of values for this data type is from -2 exp (31) to 2 exp (31) -1. |
| **Direct representation** | A method for representing variables in the PLC program from which the allocation to the logical memory location can be derived directly and, indirectly, the physical memory location. |
| **Document window** | A window within an application window. Several documents can be open simultaneously in an application window. However only one document window can be active. Document windows in Concept are, e.g., sections, the message window, the reference data editor and the PLC configuration. |
| **Duration literal** | Permitted units for durations (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or combinations thereof. The duration must be identified by the prefix t#, T#, time# or TIME#. The "Overflow" of the maximum value unit is permitted; e.g. the entry T#25H15M is permitted.<br><br>Example<br>t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS |

## E

| | |
|---|---|
| **EN / ENO (enable / fault signalling)** | If the value of EN is equal to "0", when the FFB is called the algorithms that are defined by the FFB are not carried out and all outputs retain their previous value. The value of ENO is automatically set to "0" in this case. If the value of EN is equal to "1", when the FFB is called the algorithms that are defined by the FFB are carried out. Following the error-free execution of these algorithms the value of ENO is automatically set to "1". If an error occurs during the execution of these algorithms, ENO is automatically set to "0" . The output behaviour of the FFBs depends on whether the FFBs are called without EN/ENO or with EN=1. If the indication of EN/ENO is enabled, it is imperative that the EN input is connected. Otherwise the FFB is never carried out. The projection of EN and ENO is enabled or disabled in the function block properties dialog box. The dialog box is opened using the menu commands **Objects** → **Properties...** or by double clicking the FFB. |
| **Equipment** | A unit/device, e.g. pump, lifter or motor is termed an item of equipment. The equipment forms the central projection element. It is characterised by the configuration (manual/switch flag), actions, and reactions. Equipment can be combined into equipment groups. |
| **Equipment group** | The individual items of equipment configured are combined into an equipment group based on logical or technical considerations. This facilitates technical or machine-orientated representation of the equipment at Handtableau program runtime. In addition, the connection of the individual items of equipment during program runtime is performed via the selection of the equipment groups. |
| **Error** | If an error is detected during the processing of an FFB (e.g. incorrect input values or a timing error), an error message is displayed; you can view the error using the menu command **Online** → **Event Display...**. In FFBs the ENO output is set to "0". |
| **Evaluation** | The process by means of which a value for a function or for the outputs of a function block is determined during program execution. |

## F

| | |
|---|---|
| **FFB (Functions/ Function Blocks)** | Collective term for EFB (basic functions/function blocks) and DFB (derived function blocks) |

| | |
|---|---|
| **Field variables** | Variables to which a defined derived data type is allocated with the aid of the keyword ARRAY (field). A field is a collection of data elements of the same data type. |
| **Function (FUNK)** | A program organisation unit  that on execution provides exactly one data element. A function has no internal status information. Multiple calls of the same function with the same input parameter values always produce the same output values.<br>You will find details on the graphic form of function calls in the definition "function block (instance)". Contrary to function block calls, function calls have only one unnamed output, as its name is the same as the name of the function itself. In FBD each call is identified by a unique number via the graphic block; this number is generated automatically and cannot be changed. |
| **Function block (instance) (FB)** | A function block is a program organisation unit that calculates values for its outputs and internal variable(s) in accordance with the functionality defined in its function block type description, when it is called as a specific instance. All values for the outputs and internal variables of a specific function block instance are retained from one call of the function block to the next. Multiple calls of the same function block instance with the same arguments (values for input parameters) do not therefore necessarily produce the same output value(s).<br>Each function block instance is represented graphically using a rectangular symbol. The name of the function block type is given in the centre of the top of the rectangle. The name of the function block instance  is also given at the top, however outside of the rectangle. This is automatically generated on the creation of an instance, however it can be modified by the user as required. Inputs are displayed on the left, outputs on the right of the block. The names of the formal input/output parameters are displayed inside the square at the appropriate point.<br>The above description of the graphic representation is in principle also valid for function calls and for DFB calls. Differences are described in the corresponding definitions. |
| **Function block language (FBD)** | One or more sections that contain the graphically displayed network of functions, function blocks and links. |
| **Function block type** | A language element comprising: 1. The definition of a data structure divided into input, output, and internal variables; 2. A set of operations that are performed with the elements in the data structure when an instance of the function block type is called. This set of operations can be formulated either in one of the IEC languages (DFB type) or in "C" (EFB type). One function block type can be multiply instanced (called). |

| | |
|---|---|
| **Function counter** | The function counter is used to uniquely identify a function in a program or DFB. The function counter cannot be edited and is assigned automatically. The function counter always has the structure: .n.m<br><br>n = number of the section (sequential number)<br>m = number of the FFB object in the section (sequential number) |

## G

| | |
|---|---|
| **Generic literals** | If it is unimportant to you which data type a literal has, simply enter the value for the literal. In this case, Concept automatically assigns a suitable data type. |
| **Global derived data types** | Global derived data types are available in each Concept project and are saved in the DFB directory directly under the Concept directory. |
| **Global DFBs** | Global DFBs are available in each Concept project and are saved in the DFB directory under the Concept directory. |
| **Groups (EFBs)** | Some EFB libraries (e.g. the IEC library) are subdivided into groups. This makes it considerably easier to locate EFBs in large libraries. |

## I

| | |
|---|---|
| **I/O installation list** | In the I/O installation list the I/O and expert modules of the different central units are configured. |
| **Icon** | Graphic representation of different objects in Windows, e.g. drives, application programs and document windows. |
| **IEC 1131-3** | International standard: Programmable Controllers - Part 3: Programming Languages. March 1993. |

**IEC naming convention (identifier)**

An identifier is a sequence of letters, digits and underscores that must start with a letter or an underscore (e.g. name of the function block type, an instance, a variable or a section). Letters from national character sets (e.g.: ö,ü, é, õ) can be used, except in project and DFB names.
Underscore characters are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as different identifiers. Several leading underscore characters and sequences of several underscore characters are not allowed.
Identifiers must not contain any spaces. Upper and lower case is not significant; e.g.. "ABCD" and "abcd" are interpreted as the same identifier.
Identifiers are not allowed to be keywords.

**Initial value**

The value assigned to a variable at program start. The value is assigned in the form of a literal.

**Input bits (1x reference)**

The 1/0 state of input bits is controlled by the process data that passes from an input device to the CPU.

**Note:** The x that follows the first digit of the reference type represents a five-digit memory location in the user data memory, e.g. the reference 100201 signifies an input bit at address 201 of the signal memory.

**Input parameter (input)**

Passes the associated argument during a FFB call.

**Input words (3x references)**

An input word contains information that stems from an external source and is represented by a 16 bit number. A 3x register can also contain 16 sequential input bits that have been read into the register in binary or BCD (binary coded decimal) format. Note: The x that follows the first digit of the reference type represents a five-digit memory location in the user data memory, e.g. the reference 300201 signifies a 16-bit input word at address 201 of the signal memory.

**Instance name**

An identifier that belongs to a specific function block instance. The instance name is used to uniquely identify a function block in a program organisation unit. The instance name is generated automatically, however it can be edited. The instance name must be unique in the entire program organisation unit; here a differentiation is not made between upper and lower case. If the name entered already exists, you will be warned and must choose a different name. The instance name must comply with the IEC naming conventions, otherwise an error message is displayed. The automatically generated instance name always has the structure: FBI_n_m

FBI = function block instance
n = number of the section (sequential number)
m = number of the FFB object in the section (sequential number)

| | |
|---|---|
| **Instancing** | The generation of an instance. |
| **INT** | INT stands for the data type "integer". The entry can be an integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 16 bits. The range of values for this data type is from -2 exp (15) to 2 exp (15) -1. |
| **Integer literals** | Integer literals are used for entry of integer values in the decimal system. The values can have a sign (+/-) in front. Individual underscore characters ( _ ) between the digits are not significant.<br><br>Example<br>-12, 0, 123_456, +986 |

## K

| | |
|---|---|
| **Keywords** | Keywords are unique combinations of characters that are used as special syntactical elements as defined in the Appendix B of IEC 1131-3. All keywords that can be used in the IEC 1131-3 and thus in Concept are listed in Appendix C of IEC 1131-3. These listed keywords are not permitted to be used for any other purpose, e.g., not as variable names, section names, instance names, etc. |

## L

| | |
|---|---|
| **Language element** | Each basic element in one of the IEC programming languages, e.g. a step in SFC, a function block instance in FBD or the initial value of a variable. |
| **Library** | Collection of software objects that are intended for reuse during the programming of new projects, or even for the creation of new libraries. Examples are the library for the basic function block types.<br>EFB libraries can be subdivided into groups. |
| **Link** | A control or data flow link between graphic objects (e.g. function blocks in the FBD editor) within a section, represented graphically as a line. |

| | |
|---|---|
| **Literals** | Literals are used to provide inputs on FFBs, transition conditions, etc directly with values. These values cannot be overwritten by the program logic (write protected). Here a differentiation is made between generic and classified literals.<br>In addition, literals are used to assign a value to a constant or an initial value to a variable.<br>Entry is made as base 2 literal, base 8 literal, base 16 literal, integer literal, real literal or real literal with exponent. |
| **Local derived data types** | Local derived data types are only available in a single Concept project and its local DFBs and are stored in the DFB directory under the project directory. |
| **Local DFBs** | Local DFBs are only available in a single Concept project and are saved in the DFB directory under the project directory. |
| **Located variable** | Located variables are assigned to a memory address (reference addresses 0x, 1x, 3x,4x). The value of this variable is saved in the signal memory and can be changed online using the reference data editor. These variables can be addressed using their symbolic names or using their reference address.<br><br>All inputs and outputs for the PLC are connected to the signal memory. The access of the program to peripheral signals that are connected to the PLC is only performed using located variables. External accesses via Modbus or Modbus Plus interfaces on the PLC, e.g., from information display systems, are also possible via located variables. |

## M

| | |
|---|---|
| **Manual flag** | If the manual control variable is configured as a manual flag, the variable is set to "1" with the rising edge of the signal as long as the corresponding control key is pressed. |
| **Multielement variables** | Variables to which a derived data type defined using a STRUCT or ARRAY is assigned.<br>A differentiation is made here between field variables and structured variables. |

## O

| | |
|---|---|
| **OFS (OPC Factory Server)** | The OPC Factory Server (OFS) is the interface between the individual software components for the display of information, Handtableau, programming and application program. A prerequisite for the use of this interface is that all users are capable of communicating using OPC (OLE for Process Control). The OFS runs in the background. |
| **Output parameter (output)** | A parameter with which the result(s) of the evaluation of a FFB is/are fed back. |
| **Output/flag bits (0x references)** | An output/flag bit can be used to control real output data using an output module in the control system, or to define one or more discrete outputs in the signal memory. Note: The x that follows the first digit of the reference type represents a five-digit memory location in the user data memory, e.g. the reference 000201 signifies an output or flag bit at address 201 of the signal memory. |
| **Output/flag words (4x references)** | An output/flag word can be used for the storage of numerical data (binary or decimal) in the signal memory, or also for the transmission of data from the CPU to an output module in the control system. Note: The x that follows the first digit of the reference type represents a five-digit memory location in the user data memory, e.g. the reference 400201 signifies a 16-bit output/flag word at address 201 of the signal memory. |

## P

| | |
|---|---|
| **PLC** | Programmable controller |
| **Program** | The topmost program organisation unit. A complete program is loaded on a single PLC. |
| **Program cycle** | A program cycle comprises the reading of the inputs, the processing of the program logic and the output of the outputs. |
| **Program organisation unit** | A function, a function block, or a program. This term can relate to either a type or an instance. |

| | |
|---|---|
| **Programming unit** | Hardware and software that supports programming, projecting, testing, commissioning and faultfinding in PLC applications, as well as in decentral system applications to facilitate source documentation and archiving. The programming unit can, amongst other tasks, be used for the display of process information. |
| **Project** | General term for the topmost level of a software tree structure that defines the superordinate project name of a PLC application. After definition of the project name, you can save your system configuration and your control program under this name. All data that is produced during the creation of the configuration and the program belong to this superordinate project for this special automation task. General term for the complete set of programming and projecting information in the project database, this represents the source code that describes the automation of a system. |
| **Project database** | The database in the programming unit that contains the projection information for a project. |

## R

| | |
|---|---|
| **Reaction signals** | Reaction signals represent the states of the process reactions assigned and additional signals such as, e.g., interlocks or limit switches. |
| **REAL** | REAL stands for the data type "floating point number". The entry is made as a real literal or as a real literal with exponent. The length of the data elements is 32 bits. The range of values for variables of this data type is from 8.43E-37 to 3.36E+38. |
| **Real literals** | Real literals are used for entry of floating point values in the decimal system. Real literals are identified by the entry of the decimal point. The values can have a sign (+/-) in front. Individual underscore characters ( _ ) between the digits are not significant.<br><br>Example<br>-12.0, 0.0, +0.456, 3.14159_26 |

**Real literals with exponent**

Real literals with exponent are used for entry of floating point values in the decimal system. Real literals with exponent are identified by the entry of the decimal point. The exponent defines the power of ten with which the preceding number is to be multiplied to obtain the value to be represented. The values can have a sign (+/-) in front. Individual underscore characters ( _ ) between the digits are not significant.

Example
-1.34E-12 or -1.34e-12
1.0E+6 or 1.0e+6
1.234E6 or 1.234e6

**Reference**

Each direct address is a reference that starts with a code that defines whether the address is an input or output, and whether the data is a bit or word. References that start with the code 6 represent registers in the extended memory of the signal memory.
0x range = output/flag bits
1x range = input bits
3x range = input words
4x range = output/flag words
6x range = register in the extended memory

**Note:** The x that follows the first digit of each reference type represents a five-digit memory location in the user data memory, e.g. the reference 400201 signifies a 16-bit output or flag word at address 201 of the signal memory.

**Register in the extended memory (6x reference)**

6x references are flag words in the extended memory of the PLC. They can only be used for LL984 application programs and only if a CPU 213 04 or CPU 424 02 is used.

**Runtime errors**

Errors that occur during the processing of a program in the PLC, in SFC objects (e.g. steps) or FFBs. These are, e.g., value range overflows for numbers or timing errors for steps.

## S

| | |
|---|---|
| **Section** | A section can, e.g., be used to describe the method of operation of a technical unit, such as a motor.<br>A program or DFB comprises one or more sections. Sections can be programmed using the IEC programming languages FBD and SFC. Within a section only one of the stated programming languages is permitted to be used.<br>Each section has its own document window in Concept. However, for reasons of clarity, it is sensible to divide a very large section into several smaller sections. The scroll bar is used to scroll within the section. |
| **Signal memory** | The signal memory is the memory area for all parameters that are addressed in the application program using references (direct representation). For example, input bits, output/flag bits, input words, and output/flag words are contained in the signal memory. |
| **Structured variable** | Variables to which a derived data type defined using a STRUCT (structure) is assigned.<br>A structure is a collection of data elements with, in general, different data types (basic data types and/or derived data types). |
| **Switch flag** | If the manual control variable is configured as a switch flag, the variable is switched high (to "1") on the operation of the left control key, and switched low (to "0") on the operation of the right control key. The changeover takes place with the rising edge of the signal. |

## T

| | |
|---|---|
| **TIME** | TIME stands for the data type "duration". The entry is made as a duration literal. The length of the data elements is 32 bits. The range of values for variables of this data type is from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms. |

## U

| | |
|---|---|
| **UDEFB** | User defined basic functions/function blocks<br>Functions or function blocks that have been created in the programming language C and that Concept makes available in libraries. |
| **UDINT** | UDINT stands for the data type "unsigned double integer". The entry can be an integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 32 bits. The range of values for variables of this type is from 0 to 2exp(32)-1. |
| **UINT** | UINT stands for the data type "unsigned integer". The entry can be an integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 16 bits. The range of values for variables of this type is from 0 to (2exp16)-1. |
| **Unlocated variable** | An unlocated variable is not assigned a signal memory address. In this way they do not occupy any signal memory addresses. The value of this variable is saved internally in the system and can be changed using the reference data editor. These variables are only addressed using their symbolic names.<br><br>Signals that do not require direct access to the peripherals, e.g. intermediate results, system flags, etc., should preferably be declared as unlocated variables. |

## V

| | |
|---|---|
| **Variables** | Variables are used for the exchange of data within sections, between several sections and between the program and the PLC.<br>Variables comprise at least one variable name and a data type.<br>If a variable is assigned a direct address (reference), the term located variable is used. If a variable is not assigned a direct address, the term unlocated variable is used. If a derived data type is assigned to the variable, the term multielement variable is used.<br>In addition, there are also constants and literals. |

## W

| | |
|---|---|
| **Warning** | If a critical state is detected during the processing of a FFB or step (e.g. critical input values or time limit exceeded), a warning is displayed that you can view using the menu command **Online** → **Event display...**. In FFBs the ENO output remains at "1". |
| **WORD** | WORD stands for the data type "sequence of 16 bits". The entry can be a base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 16 bits. A range of numerical values cannot be assigned to this data type. |

# Index